HELSINGIN KAUPPAKORKEAKOULU

Kvantitatiivisen suunnittelun
koulutusohjelma

# TIETOKONEAVUSTEINEN SYSTEEMITYÖ (TAS)



Liiketaloustiede: Systeemien ja
tietojenkäsittelyn
pro gradu -tutkielma
Jukka Iivonen
Kevätlukukausi 1991

---

Hallinnon ja systeemien laitoksen laitos-
neuvoston kokouksessa 19 / 4 1991 hyväksytty
arvosanalla cum laude approbatur

MARKKU SÄÄKSJÄRVI
TkT Markku Sääksjärvi

JUKKA HEIKKILÄ
Ekonomi (uusi) Jukka Heikkilä

HELSINGIN KAUPPAKORKEAKOULU
Tietojenkäsittely
pro gradu -tutkielma
Jukka Iivonen

**TIIVISTELMÄ**

15.4.1991

# TIETOKONEAVUSTEINEN SYSTEEMITYÖ (TAS)

## Tutkimuksen tavoitteet

Tutkimuksessa pyrittiin selvittämään tietokoneavusteisen systeemityön (TAS) keskeinen viitekehys. Tämän pohjalta pyrittiin esittämään TAS-välineen tärkeimmät valintakriteerit.

Tutkimuksella oli tavoitteena myös selvittää HKKK:n systeeminsuunnittelun opetuksen tämän hetkinen tila ja tehdä mahdolliset parannusehdotukset tutkimuksen aikana selvitetyn alan uusimman viitekehyksen pohjalta.

## Lähdeaineisto

Lähdeaineistona tutkimuksessa käytettiin alan uusinta kirjallisuutta sekä julkaistuja artikkeleita.

Lähdeaineistona olivat myös tekijän omat havainnot HKKK:n tietojenkäsittelyn opettajana ja assistenttina.

## Tulokset

Keskeisenä tuloksena tutkimuksesta saatiin TAS-välineen valintakriteerit erilaisille organisaatioille ja erilaisten sovellusten kehittämiseen. Tutkimus antoi myös kattavan viitekehyksen TAS:n opetukselle HKKK:ssa.

Lisäksi tuloksena tutkimuksesta saatiin parannusehdotus HKKK:n systeemityön koko opetusrakenteeseen ja kurssien keskinäiseen integroimiseen.

Avainsanat Tietokoneavusteinen systeemityö (TAS), systeemityömenetelmä, vaihejako, kuvauskanta.

SISÄLLYSLUETTELO

KUVIEN LUETTELO

**TAULUKKOJEN LUETTELO**

# 1. JOHDANTO

## 1.1 Tausta

Tietosysteemien rakentamisen tueksi on kehitetty erilaisia malleja ja menetelmiä. Niiden käytössä on ollut kuitenkin vaikeuksia. Myös itse rakentamistyöhön on kehitetty apuvälineitä, mutta ne ovat keskittyneet työn toteutusvaiheeseen. Tällaisia apuvälineitä ovat mm. sovelluskehittimet, koodigeneraattorit ja neljännen sukupolven ohjelmointikielet. Suunnittelu on sitävastoin ollut vaikeimmin tietokoneella autettavissa. Suunnittelun kalleus sekä vaikeus, hitaus, toisaalta mikrotietokoneiden antamat mahdollisuudet ovat johtaneet menetelmien ja kokonaisvaltaisten, koko ohjelmiston elinkaaren kattavien apuvälineiden kehittämiseen. Tällaisista systeeminsuunnittelun apuvälineistä on käytetty nimitystä CASE (Computer-Aided Software Engineering) eli suomeksi TAS (Tietokoneavusteinen Systeemityö).[1]

TAS alkoi kehittyä USA:ssa 1980-luvun alussa. Systeemityössä oli ajauduttu tilanteeseen, jossa uusien sovellusten kehitysjättämä oli useita vuosia[2]. Näin liiketoiminnan kehittymisen edellyttämien uusien systeemien kehitystyöhön jäi täysin riittämättömästi aikaa. Systeemityön tehostamiseen oli siis mitä ilmeisin tarve. Tietokoneavusteinen systeemityö on tunnustettu tärkeäksi mahdollisuudeksi kohottaa sekä systeemityön tuottavuutta että laatua.

Tietokoneavusteisen systeemityön opetus alkoi Helsingin Kauppakorkeakoulussa (HKKK) keväällä 1989 erillisenä Systeeminsuunnittelun CASE-kurssina. HKKK:ssa on haluttu pitää systeeminsuunnittelun opetus ajanmukaisena. Koululle on hankittu kaksi erilaista TAS-välinettä:

---

[1] SYTYKE, TAS-raportti s.1

[2] Martin 1982, s. 4

DEFT ja IEW. DEFT toimii Macintosh-ympäristössä ja IEW MS-DOS-ympäristössä. DEFTin taustalla ei ole mitään tiettyä systeemityömenetelmää, eikä se kata kuin osan systeemityön vaiheista. DEFTin etuna on kuitenkin sen liittymät useisiin sovelluskehittimiin, kuten INGRESiin ja ORACLEen, joiden tietokantamääritykset saadaan generoitua suoraan DEFTin kuvauksista. IEW on puolestaan systeemityön kannalta täydellisempi sisältäen osat yrityksen toiminnallisen tason suunnittelusta aina yksittäisen tietojärjestelmän tietojen ja toimintojen mallintamiseen. IEW perustuu Martinin Information Engineering -menetelmään, joskin IEW:llä voidaan käyttää muitakin menetelmiä[1].

## 1.2 Tutkimuksen päämäärä ja tavoitteet

Tutkimuksen päämääränä on selvittää TAS:n keskeinen viitekehys TAS-välineen valintaa varten. Tämän viitekehyksen on tarkoitus samalla antaa suuntaviivat systeemityön ja etenkin TAS:n opetukselle HKKK:ssa.

Tutkimuksen tavoitteena on perehtyä TAS:hön sekä sen kehitysnäkymiin kirjallisuuden perusteella ja kuvata se selkeästi ja kattavasti.

Toisena tavoitteena on pyrkiä arvoimaan HKKK:n systeemityön opetuksen tila ja tehdä parannusehdotukset tutkimuksen viitekehyksen pohjalta.

## 1.3 Rajaus

Tämä tutkimus käsittelee TAS:tä ja systeemityömenetelmiä pääasiassa vain informaatiojärjestelmien kannalta. Systeemityön opetus käsittää vain opetuksen HKKK:ssa.

---

[1] SYTYKE TAS-Raportti s. 131-132

## 1.4 Keskeiset käsitteet

**Vaihejako** — Tapa jäsentää systeemityö peräkkäisiksi erityyppisiksi työsuorituksiksi, jotka työn kuluessa voivat toistua.[1]

**Systeemityö** — Tietosysteemien kehittäminen ja kunnossapito.[2] Systeemityö etenee yleensä vaihejaon perusteella.

**Tietokoneavusteinen systeemityö (TAS)** — Systeemityön tai sen osan suorittaminen yhtä tai useampaa tietokoneohjelmaa (TAS-välinettä) hyväksikäyttäen.

**Kuvauskanta (Dictionary/Repository/ Encyclopedia)** — Tietovarasto, johon tallennetaan TAS:n aikana kuvauksia ja määrityksiä kehitettävästä tietojärjestelmästä. Kuvauskantaan tallennetaan myös tiedot kuvausten keskinäisistä yhteyksistä.

TAS:hön liittyy myös muita keskeisiä käsitteitä, joita McCluren mukaan ovat:[3]

**TAS-väline** (CASE tool) — Ohjelma, joka automatisoi osittain tietyn osan systeemityöstä.

**TAS-välineistö (CASE toolkit)** — Kokoelma yhteensopivia TAS-välineitä, jotka yhdessä automatisoivat kokonaan tietyn vaiheen systeemityöstä.

---

[1] ATK-sanakirja

[2] ATK-Sanakirja

[3] McClure, 1989 s. 16.

**TAS-ohjelmisto**
(CASE workbench)

Kokoelma yhteensopivia TAS-välineitä, jotka yhdessä automatisoivat kaikki systeemityön vaiheet, esitutkimuksesta ylläpitoon.

**TAS-menetelmäohjelmisto**
(CASE methodology companion)

Kokoelma yhteensopivia TAS-välineitä, jotka pohjautuvat tiettyyn systeemityömenetelmään. Ne automatisoivat osin tai kokonaan ko. menetelmän eri vaiheet.

**TAS-ympäristö**
(CASE hardware platform)

Yksi-, kaksi- tai kolmekerroksinen järjestelmäarkkitehtuuri sen mukaan, käytetäänkö TAS-välineitä standalone mikrotietokoneelta, mikro- ja minikonelta vai mikro-,mini- ja keskuskoneelta.

**TAS-järjestelmä**
(CASE system)

Kokoelma yhteensopivia TAS-välineitä, joilla on sama käyttöliittymä, ja jotka toimivat tietyssä TAS-ympäristössä.

## 2. SYSTEEMITYÖN MENETELMISTÄ

Tietosysteemien koon kasvaessa ja monimutkaisuuden lisääntyessä on välttämätöntä käyttää jotain loogista menetelmää systeemien määrittelyssä ja suunnittelussa. Suunnittelu- ja toteutustyö saattaa olla sekä paikallisesti että maantieteellisesti hajautettua. Tällöin on suunnittelutyötä ohjaava ja integroiva suunnittelumenetelmä välttämätön, jotta eri osista saataisiin kuvattua toteutettava tietosysteemi, joka täyttää sille asetetut toiminnalliset tavoitteet.[1]

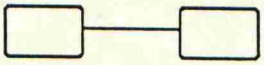
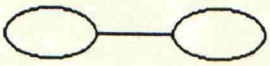
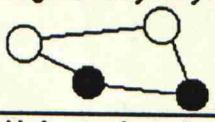
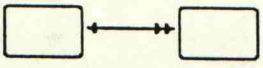
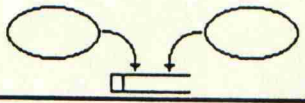
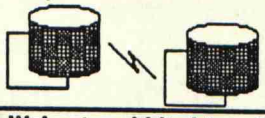
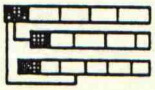
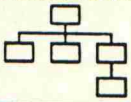
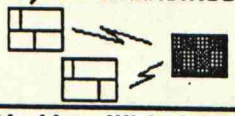
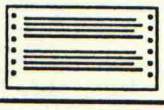
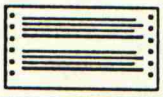
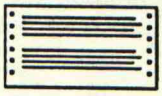
Zachmanin mielestä ei ainakaan vielä ole olemassa suunnittelumenetelmää, joka yksinään kattaisi koko tietosysteemin kuvaustarpeet. Tosin erilaisista menetelmistä voidaan yhdessä muodostaa kattava kokoelma menetelmiä suunnittelu- ja rakentamistyötä ohjaamaan. Zachman esittääkin tietosysteemien kehittämisen viitekehyksen havainnollistaen siinä eri näkökulmat, joista tietosysteemejä hänen mielestään on tarpeellista tarkastella. Eri tarkastelunäkökulmiin sopivat eri kuvausmenetelmät.

Tietosysteemien tieto-, toiminto- ja verkkokuvauksia voidaan tarkastella johdon, käyttäjän, suunnittelijan, ohjelmoijan sekä tietokoneen näkökulmista (Kuva 1.). Esimerkiksi johto käsittää "työntekijän" fyysiseksi olennoksi, mutta ohjelmoija tulkitsee "työntekijän" tietueeksi tietokannassa, jolla on tietty avain, jolla se yksilöidään. Johdon määritelmä "työntekijästä" on verbaalinen, hyvin vapaa tulkinta. Ohjelmoijan määritelmä on puolestaan toteutusvälineriippuvainen, tiukka määritelmä. Tarvitaan siis kaksi erillistä kuvausta, jotta molemmat tulkinnat saataisiin kuvattua.

---

[1] Zachman, 1987 s. 276

Kuva 1. TIETOSYSTEEMIEN ERI TARKASTELUNÄKÖKULMAT

| | Tietomääritykset | Toimintomääritykset | Arkkitehtuuri | Tarkastelu-perspektiivi |
|---|---|---|---|---|
| Järjestelmän yleiskuva | Liiketoiminnalle tärkeät tiedot  | Liiketoiminnalle tärkeät toiminnot  | Liiketoiminnan sijainti  | Johto |
| Liiketoiminta-malli | Kohde-yhteyskaavio  | Toimintojen yhteydet  | Logistiset yhteydet  | Käyttäjä |
| Tietojärjestelmämallit | Kohdemalli  | Tietovirtakaavio  | Hajautuskaaviot  | Suunnittelija |
| Yksityiskohtaiset mallit | Tietokantakuvaukset  | Rakennekaaviot  | Järjest. arkkitehtuuri  | Ohjelmoija |
| Tietojärjestelmän määritykset | Tietokantamääritykset  | Ohjelmakoodi  | Verkkomääritykset  | Tietokone |
| Todellinen järjestelmä | Tiedot | Toiminnot | Kommunikointi | |

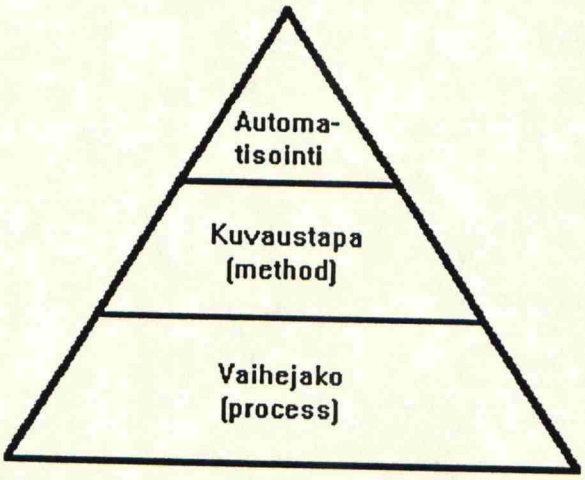
Lähde: Tekijän mukaelma kuvasta Zachman, 1987 s. 285

Zachman esittää siis 15 eri näkökulmaa, joista tietosysteemejä on mahdollista tarkastella. Manuaalisesti tehtynä 15 eri kuvauksen keskinäinen eheys on kuitenkin

vaikea hallita. Eri tason kuvaukset ovat yleensä eri ihmisten (johto/suunnittelija/ohjelmoija) tekemiä, joten informaatiokatkoksia varmastikin esiintyy. Tietokoneavusteisesti tehtynä kuvauksien keskinäinen eheys on kuitenkin huomattavasti helpompi ylläpitää ja se voidaan nopeasti tarkastaa. TAS-välineillä tehtyjä kuvauksia voidaan myös helposti käyttää apuna uusien sovellusten suunnittelussa ja toteutuksessa. Tietokoneavusteisuudella onkin näin suuri vaikutus tietosysteemien laadun paranemiseen, kehitystyön nopeutumiseen sekä toisaalta kehitys- ja ylläpitokustannusten alenemiseen.

TAS-välineen keskeisin ominaisuus on kuvausmenetelmä ja sen automatisointi. Kuvausmenetelmä koostuu kuvaustekniikasta (method), jonka avulla tarvittavat kaaviot ja määrittelyt tehdään ja vaihejakomallista (process), joka ohjaa tätä työtä ja toisaalta valvoo, että kaikki tarvittavat systeemin kuvaukset tehdään. Näiden kuvausten tekoa voidaan TAS-välineellä automatisoida ja niiden avulla voidaan jopa ohjelmakoodi generoida automaattisesti (Kuva 2).[1]

Kuva 2. TAS-MENETELMÄN MALLI




Lähde: Charette, 1988 s.39

[1] Charette, 1988 s. 39

## 2.1 Systeemityömenetelmän valinta

### 2.1.1 Valinta tietojenkäsittelyn kehittymismallin perusteella

TAS-välineen tärkein ominaisuus on myös Finkelsteinin[1] mielestä se menetelmä, jota väline käyttää. Menetelmän tulee sopia yrityksen toimintaan. Hän esittää Nolanin (Nolan, 1979) tietojenkäsittelyn kasvumallin vaiheet pohjaksi sopivimman TAS-menetelmän löytämiseksi. Nolanin mallin vaiheet ovat (Kuva 3):[2]

**1. Käynnistys**
Tietokoneilla käytetään lähinnä yksittäisiä operationaalisia toimintoja tukevia sovelluksia, kuten kirjanpito, varastonvalvonta ja laskutus. Niillä pyritään vähentämään näiden toimintojen kustannuksia.

**2. Laajeneminen**
Käynnistysvaiheen sovellusten onnistuneen toiminnan kannustamana käyttäjät vaativat uusia sovelluksia ja käynnistysvaiheesta siirrytään laajenemiseen. Tietojenkäsittelyn annetaan kehittyä melko vapaasti. Tietojenkäsittely leviää nopeasti kaikkialle operatiivisiin järjestelmiin. Organisaatiolta puuttuu kuitenkin järjestelmä, jolla se valvoisi ja ohjaisi vielä kokemattomien suunnittelijoidensa ja ohjelmoijiensa työtä. Huonosti suunniteltujen ja dokumentoitujen ohjelmien ylläpito vie myöhemmin jopa 70-80% suunnittelijoiden ja ohjelmoijien työajasta.

**3. Valvonta**
Laajeneminen karkaa tietojenkäsittelyosaston hallinnasta. Suurin osa suunnittelijoiden ja ohjelmoijien työajasta kuluu vanhojen sovellusten ylläpitotyöhön, joten uusille sovelluksille syntyy kehitysjättämää,

---

[1] Finkelstein 1989, s.47.

[2] Nolan 1979, s. 116-120

joka on pahimmillaan vuosia. Tietojenkäsittelyn hallinto muuttuu tietokoneiden hallinnasta yrityksen tietoresurssien hallinnoksi. Otetaan käyttöön sovellusten hallintamenetelmiä, dokumentointia parannetaan ja sovelluksia rakennetaan uudelleen. Tietojenkäsittelyn organisoinnissa tehdään myös uudistuksia.

**4. Yhdentyminen**

Tietojenkäsittelyosaston uudelleen organisoinnin jälkeen otetaan tietokanta- ja tietoliikennetekniikat käyttöön useilla yrityksen toiminnalle tärkeillä sovellusalueilla, kuten kirjanpito sekä tilaustenkäsittely ja varastonvalvonta. Käyttäjät saavat myös ensimmäistä kertaa tehokasta tukea atk-osastolta laitteiden ja ohjelmien käytössä.

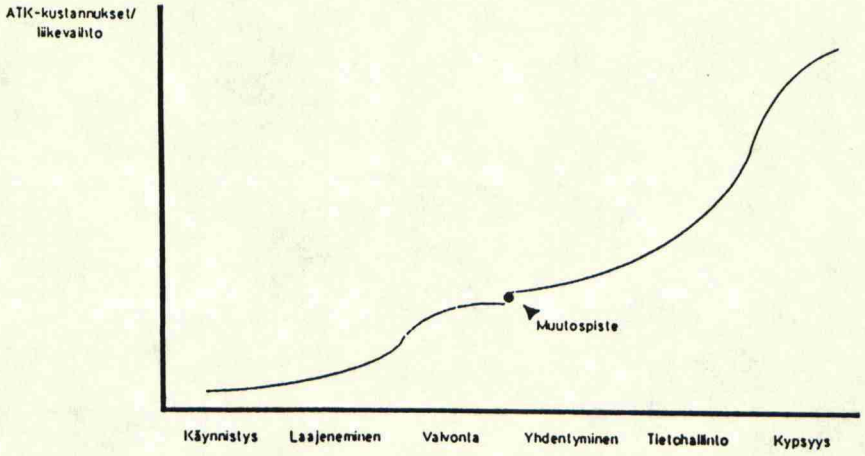
**5. Tietohallinto**

Yrityksen tietoresurssia hallitaan samoin kuin kahta muutakin yrityksen avainresurssia, rahoitusta ja henkilöstöä. Tietojenkäsittely nähdään olennaisena apuna yrityksen strategioiden toteuttamisessa. Tietohallinto on olennainen osa yritystoimintaa.

**6. Kypsyys**

Strategiset tiedot ovat ylimmän johdon käytössä päätöksentekoa tukemassa. Sovellusportfolio ja sen rakenne kuvaa organisaatiota ja sen tietovirtoja kokonaisuudessaan. Tietokannoissa olevat tiedot ovat tehokkaassa käytössä johdon päätöksenteon tukena. Vain harvat organisaatiot saavuttavat tämän tietojenkäsittelyn kypsyyden tason.

Kuva 3. TIETOJENKÄSITTELYN KEHITYSVAIHEET NOLANIN MUKAAN




Lähde: Nolan, 1979

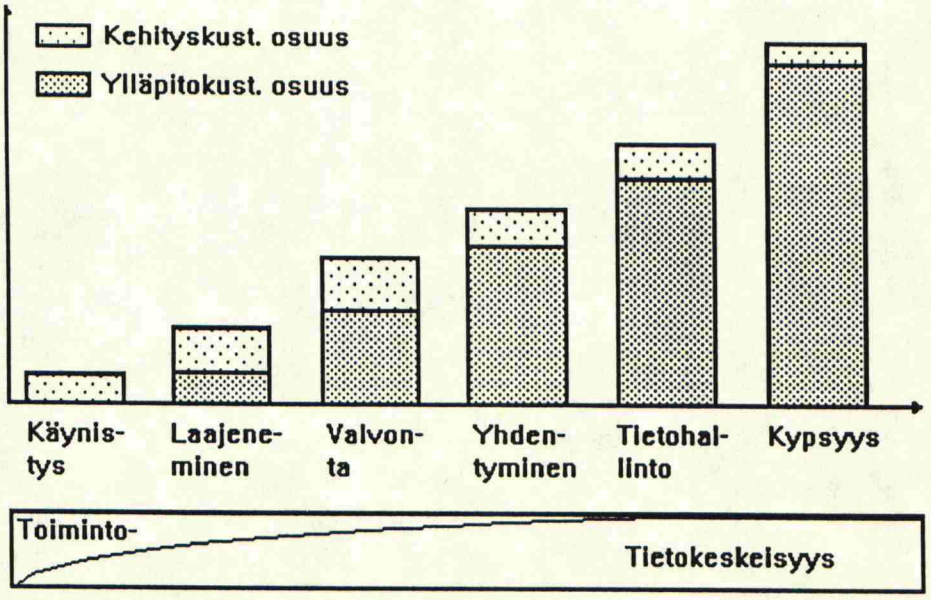
Nolanin kasvumallissa kaksi ensimmäistä vaihetta ovat Finkelsteinin mukaan toimintokeskeisiä ja neljä viimeistä puolestaan tietokeskeisiä. Vasta kolmannessa vaiheessa (valvonta) huomataan tietojen yhteiskäytön ja sovellusintegraation tarve. Aikaisemmissa kehitysvaiheissa on keskitytty yksittäisten sovellusten vaatimiin erillisiin tietoihin.

Kaksi ensimmäistä toimintokeskeistä kehitysvaihetta keskittyvät siis yksittäisiin sovelluksiin eivätkä tietoihin. Sovelluksia kehitetään käyttämällä toimintokeskeisiä menetelmiä, kuten Yourdonin SA/SD ja Gane/-Sarson SA.

Myöhemmät tietojenkäsittelyn kehitysvaiheet ovat Finkelsteinin mukaan tietokeskeisiä. Näissä vaiheissa käytettävän suunnittelumenetelmänkin tulisi olla tietokeskeinen, kuten esimerkiksi Martinin Information Engineering. Organisaation tietojenkäsittelyn pohjaksi rakennetaan tietomalli, jonka avulla integroidaan sovellukset myöhemmissä kehitysvaiheissa. Vasta 1980-luvun puolivälin jälkeen alettiin ymmärtämään, millaisia vaikeuksia toimintokeskeisten menetelmien käyttö aiheuttaa, kun organisaatio on jo siirtynyt tietokeskeiseen vaiheeseen (vaihe 3, valvonta ja sen

jälkeen). Tällöin suurin osa tietojenkäsittelyyn käytettävissä olevista varoista kuluu ylläpitotyöhön.

Kuva 4. TOIMINTOKESKEISTEN MENETELMIEN KÄYTÖN VAIKUTUKSET TIETOJENKÄSITTELYN KEHITTYMISEN ERI VAIHEISSA.

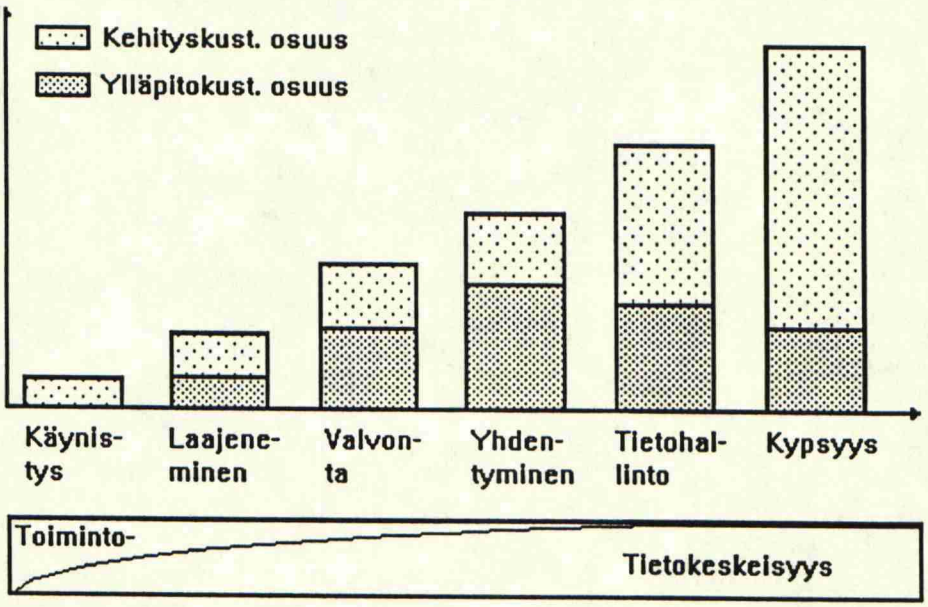



Lähde: Finkelstein 1988, s.49

Jos toimintokeskeisiä analyysi- ja suunnittelumenetelmiä käytetään tietokeskeisissä kehitysvaiheissa, organisaation on vaikea saavuttaa yhdentymis- ja tietohallintovaihe, koska ylläpitotyö kasvaa 4-6 vaiheissa hyvin suureksi (Kuva 4.). Suuren ylläpitotyön takia uusien sovellusten kehitysjättämä jatkaa kasvuaan. Monet organisaatiot eivät etenekään tämän takia valvontavaihetta pidemmälle.

Tietokeskeisen menetelmän käyttö, etenkin valvontavaiheessa ja sen jälkeen, vähentää ylläpitotyön kustannuksia ja siihen kuluvaa työmäärää. Menetelmän avulla sovellusintegraatio voidaan saavuttaa ja organisaatio voi siirtyä yhdentymis- ja tietohallintovaiheeseen. Ylläpitotyön kustannukset vähenevät tietokeskeisen menetelmän käyttöönoton jälkeen (Kuva 5). Kehitysjättämä pienenee tai sitä ei synny ollenkaan.

Kuva 5. TIETOKESKEISTEN MENETELMIEN KÄYTÖN VAIKUTUKSET TIETOJENKÄSITTELYN KEHITTYMISEN ERI VAIHEISSA




Lähde: Finkelstein 1988, s.50.

Nolanin mallin avulla arvioidaan missä tietojenkäsittelyn kehitysvaiheessa yritys on, jolloin voidaan päätellä, onko toiminto- vai tietokeskeinen systeemityömenetelmä sopivin sen tarpeisiin. Yritykselle hankittavan TAS-välineen tulee tukea tämän valitun menetelmän soveltamista.

### 2.1.2 Valinta kontingenssimallin perusteella

Yrityksessä käytettävä systeemityömenetelmä voidaan valita myös kontingenssimallin (Burns & Dennis, 1985) perusteella. Systeemityömenetelmä valitaan kehitettävien sovellusten tyypin mukaan. Sovellustyyppi kartoitetaan arvioimalla systeemityöprojektin epävarmuutta ja kehitettävän järjestelmän monimutkaisuutta kontingenssitekijöillä. Projektin epävarmuus luokitellaan kolmen kontingenssitekijän perusteella. Nämä tekijät ovat:[1]

1. **Kehitettävän järjestelmän toimintojen rakenteisuus** (Degree of Structuredness). Kohdejärjestelmän syöttö- ja tulostustiedot sekä niiden käsittelylogiikan monimutkaisuus arvioidaan.
2. **Käyttäjien ymmärrys omasta työstään** (User Task Comprehension). Arvioidaan käyttäjien kokemus ja ymmärrys työtehtävistään sekä heidän kokemuksensa sovelluskehityksestä.
3. **Kehittäjien ymmärrys käyttäjien työstä** (Developer Task Proficiency). Suunnittelijoiden kokemus kyseiseltä sovellusalueelta arvioidaan.

Kehitysprojektin monimutkaisuus arvioidaan puolestaan neljän kontingenssitekijän perusteella. Nämä tekijät ovat:

1. **Projektin koko.** Mitä enemmän aikaa kuluu projektiin sitä monimutkaisempi kehitystyö.
2. **Järjestelmän tulevien käyttäjien lukumäärä.** Mitä useampi käyttäjä järjetelmällä on sitä monimutkaisempi kehitystyö.
3. **Järjestelmän tuottaman uuden tiedon määrä.** Mitä enemmän järjestelmä tulee tuottamaan uutta tietoa sitä monimutkaisempi kehitystyö.

---

[1] Burns & Dennis, 1985 s. 21.

4. **Uuden tiedon tuottamiseen tarvittavan logiikan monimutkaisuus.** Mitä monimutkaisempaa logiikkaa joudutaan käyttämään uutta tietoa tuotettaesssa sitä monimutkaisempi kehitystyö.

Kun systemityöprojektin epävarmuus ja monimutkaisuus on arviotu kontingenssitekijöillä, voidaan kehitystyössä käytettävä systeemityömenetelmä valita taulukon 1. nelikentästä.

Taulukko 1. SYSTEEMITYÖMENETELMÄN VALINTA KONTINGENSSIMALLIN AVULLA.

| PROJEKTIN MONIMUTKAISUUS | | | |
|---|---|---|---|
| | Korkea | Perinteiseen vesiputousmalliin perustuva men. | Sekamalli |
| | Matala | Prototyyppityöskentelyyn perustuva menetelmä | Prototyyppityöskentelyyn perustuva menetelmä |
| | | Matala | Korkea |
| | | PROJEKTIN EPÄVARMUUS | |

Lähde: Burns & Dennis, 1985 s. 21

Perinteiseen vesiputousmalliin perustuvilla menetelmillä tarkoitetaan systeemityömenetelmiä, jotka etenevät vaiheittain määrittelystä suunnittelun kautta toteutukseen ja ylläpitoon. Tälläisia menetelmiä käytettäessä käyttäjillä on mahdollisuus kokeilla sovellusta vasta toteutusvaiheen jälkeen. Tällöin vasta voidaan todeta käytännössä, ovatko suunnittelijat onnistuneet toteuttamaan käyttäjien tietojenkäsittelytarpeet. Tällaista systeemityömenetelmää tulisi kontingenssimallin mukaan käyttää vain sellaisten sovellusten kehittämiseen, jotka ovat hyvin monimutkaisia käsittelylogiikaltaan, mutta joiden sovellusala on sekä käyttäjille että suunnittelijoille hyvin tuttu.

Prototyyppityöskentelyyn perustuvilla menetelmillä tarkoitetaan systeemityömenetelmiä, jotka aloittavat suunnittelutyön tulevan sovelluksen prototyypin rakentamisesta. Nämä menetelmät soveltuvat käsittelylogiikaltaan helppojen, mutta käyttäjille ja suunnittelijoille uusien sovellusalojen sovellusten toteuttamiseen, koska prototyypin avulla voidaan hyvin aikaisessa vaiheessa testata vastaavatko suunnittelijoiden näkemykset sovelluksesta käyttäjien odotuksia ja heidän todellisia tarpeitaan.

Suurien ja vaikeiden sovellusten kehittämiseen voidaan käyttää ns. sekamenetelmää, jossa käyttäjien odotukset ja tarpeet kartoitetaan prototyypin avulla. Tämän jälkeen sovellus toteutetaan perinteiseen vaihejakomalliin perustuvilla systeemityömenetelmillä.

Kontingenssimalli auttaa löytämään yritykselle sopivan systeemityömenetelmän sen perusteella minkätyyppisiä sovelluksia siellä tehdään. TAS-välineet tukevat tällä hetkellä lähinnä perinteiseen vaihejakomalliin perustuvaa systeemityötä. Niissä on vain vähäisiä protoiluominaisuuksia, joten ne ovat omimmillaan suurten ja monimutkaisten, mutta käyttäjille sekä suunnittelijoille tuttujen sovellusten toteuttamisessa. Tällaisia sovellusalueita löytyy tyypillisesti esimerkiksi pankeista.

## 2.2 Systeemityömenetelmien luokitteluja

Menetelmiä on useita erilaisia ja ne soveltuvat erilaisiin tarkoituksiin. TAS-välineet tukevat yleensä yhtä tai useampaa systeemityömenetelmää. Menetelmien luokittelu on tärkeää, jotta voitaisiin valita tietyn yrityksen sovelluskehitykseen oikeanlainen TAS-väline.

### 2.2.1 Tietosuunnittelu- vs. systeeminsuunnittelumenetelmä

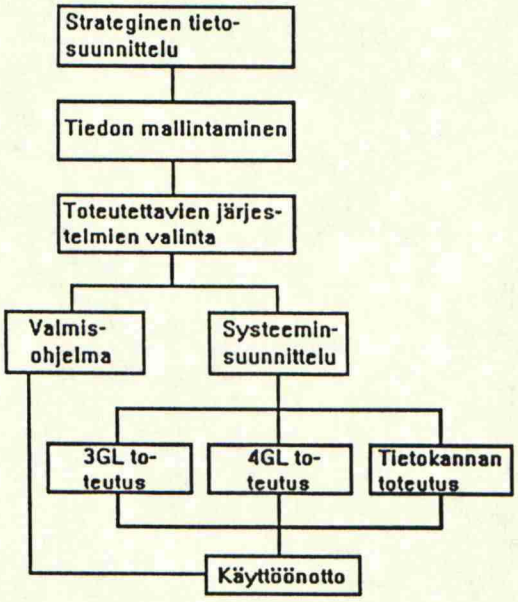
Systeemityömenetelmät voidaan jaotella **systeeminsuunnittelu**(software engineering, SE)- ja **tietosuunnittelu**menetelmiin (information engineering, IE).[1] Yksittäisen tietojärjestelmän suunnitteluvaiheessa tietosuunnittelumenetelmät (IE) soveltavat itse asiassa systeeminsuunnittelumenetelmiä (SE). Tietosuunnittelumenetelmä onkin eri systeeminsuunnittelumenetelmistä laajennettu systeemityömalli, joka on tarkoitettu pääasiassa kaupallis-hallinnollisten järjestelmien toteuttamiseen. Tosiaikajärjestelmien kehitykseen tietosuunnittelumenetelmät eivät sovellu, koska niistä ei löydy ominaisuuksia, joilla kuvataan esimerkiksi tosiaikajärjestelmien (real-time systems) ulkoisia tapahtumia (control events).

Tietosuunnittelumenetelmät (IE) ovat informaatiokeskeisiä ja aloittavat määrittelemällä yrityksen toiminnan loogisen tietomallin. Tämä kuvaa tietojen käyttöä yrityksessä. Yritys, sen liiketoiminnalliset tavoitteet sekä tietotarpeet analysoidaan ja mallinnetaan. Tietojärjestelmät määritellään ja rakennetaan tieto- ja toimintomallien pohjalta niin, että ne tukevat yrityksen liiketoiminnallisten tavoitteiden saavuttamista (Kuva 6).

---

[1] McClure, PC Tech Journal 8/88

Kuva 6. TIETOSUUNNITTELUMENETELMÄN VAIHEET




Lähde: Tekijän mukaelma kuvasta PC Tech Journal, 8/88 s. 59

Tietosuunnittelumenetelmät, kuten Martinin Information Engineering-menetelmä, eivät käsittele pelkästään yhtä tietojärjestelmää, vaan kaikkia yrityksen toimintaa ja strategioita tukevien tietojärjestelmien suunnittelua ja toteutusta sekä niiden keskinäistä integrointia. Ne käyttävät hyväkseen strategista suunnittelua ja yrityskohtaista tietomallia. Tietomallista nähdään yrityksen toiminnalle tärkeät kohteet sekä näiden väliset riippuvuudet ja yhteydet. Tietosuunnittelumenetelmän avulla voidaan tietomallista analysoida, kuinka organisaatio käyttää sille tärkeitä tietojaan. Tämän avulla voidaan johtaa organisaatiolle tietojärjestelmäarkkitehtuuri, joka tukee sen tietotarpeiden tyydyttämistä.

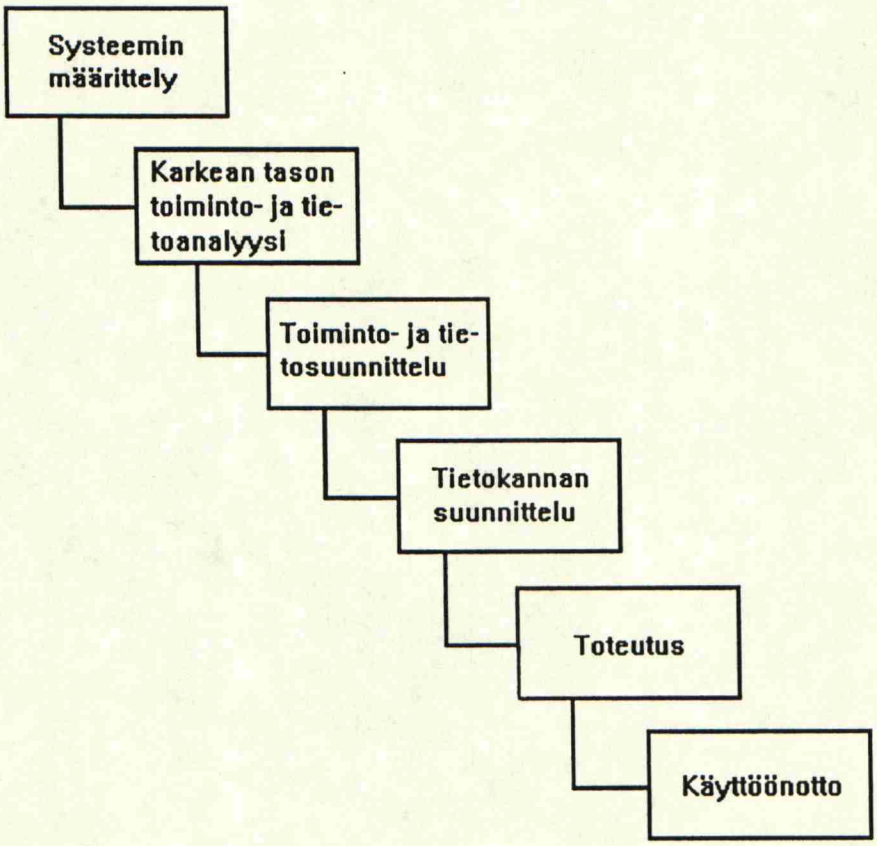
Systeeminsuunnittelumenetelmät ovat systeemityömenetelmiä, joilla suunnitellaan yksittäisiä sovelluksia. Ne voidaan luokitella toiminto-ja tietokeskeisiin menetelmiin.[1] Toimintokeskeiset menetelmät käsittelevät

[1] McClure, PC Tech Journal 8/88

toimintoja tietojärjestelmien keskeisimpänä osana. Tällaista menetelmää käytettäessä tietojärjestelmien määrittelytyö aloitetaan kuvaamalla toiminnot tietovirtakaavioiden avulla.

Tietokeskeiset menetelmät puolestaan pitävät tietojärjestelmän syöttö- ja tulostustietoja sen keskeisimpänä osana. Tietojärjestelmien määrittely aloitetaan kohdemallista. Järjestelmien toiminnot johdetaan kohdemallista siten, että jokaisen kohteen luontia, ylläpitoa ja poistoa varten määritellään tarvittavat toiminnot ja käsittelysäännöt. Lisäksi kohteista määritetään avaimet, jotka yksikäsitteisesti yksilöivät kohteen ilmentymät.

Kuva 7. SYSTEEMINSUUNNITELUMENETELMÄN VAIHEET




Lähde: Peters, 1981 s.13

Systeeminsuunnittelumenetelmissä, kuten Yourdon SA/SD (Structured Analysis/Structured Design)[1], De Marco[2] ja Gane&Sarson SA (Structure Analysis)[3], aloitetaan systeemityö yhdestä tietyn tietojärjestelmän yleisnäkemyksestä. Kyseisen järjestelmän jokainen toiminto (function) jaetaan alatoimintoihinsa, tehtävätasolle (process). Tämän jälkeen tehtävät voidaan ohjelmoida. Nämä menetelmät ovat siis toimintokeskeisiä systeeminsuunnittelumenetelmiä.

Tietokeskeisissä systeemityömenetelmissä, kuten DSSD (Data Structured Systems Development), aloitetaan suunnittelutyö yhden tietojärjestelmän tietotarpeista, jotka normalisoidaan ja tietokantojen suunnittelusta. Järjestelmän sisältämille tiedoille määritellään tämän jälkeen käsittelysäännöt, toiminnot.

### 2.2.2 Muita luokitteluja

Systeemityömenetelmät voidaan jakaa ryhmiin myös sen mukaan, millaisten tietojärjestelmien sunnitteluun ne on tarkoitettu. Tosiaikajärjestelmien (real-time systems) suunnitteluun eivät kaikki menetelmät kykene. Ne vaativat suunnittelumenetelmältä kykyä mallintaa myös ulkoiset tapahtumat (control events), jotka ohjaavat tosiaikajärjestelmiä. Joihinkin systeeminsuunnittelumenetelmiin on jälkeenpäin lisätty ominaisuuksia, joilla näitä tosiaikajärjestelmien erikoisvaatimuksia voidaan hallita (taulukko 2).[4]

Systeemityömenetelmät voidaankin jaotella myös analyysi- tai suunnittelumenetelmiin sen mukaan minkä vaiheen

---

[1] Yourdon, 1979

[2] DeMarco, 1979

[3] Gane & Sarson, 1979

[4] McClure, PC Tech Journal, 8/88 s. 55

kuvauksia ne sisältävät. Jotkut menetelmät tukevat molempia vaiheita. McClure kutsuu näitä kahtaismenetelmiksi (Dual methodologies).[1]

Suurin osa menetelmistä noudattaa ylhäältä-alas-suunnittelua (Top-Down approach). Ne aloittavat tietojärjestelmien kuvaamisen karkean tason tieto- tai toimintotarkastelulla (analyysivaihe) ja etenevät kuvaamalla nämä yhä yksityiskohtaisemmin. Jokainen toiminto hajoitetaan alatoimintoihinsa, kunnes saavutetaan toiminnon yksittäinen tehtävätaso (suunnitteluvaihe). Jokaisen tehtävän logiikka kuvataan koodausta varten.

Taulukko 2. ERI SYSTEEMITYÖMENETELMIEN LUOKITTELUJA

| | DeMarco | DSSD | Gane-Sarson | Jackson | Martin | Yourdon |
|---|---|---|---|---|---|---|
| OHJAAVUUS | | | | | | |
| Tieto | | ● | | | ● | |
| Informaatio | | | | | ● | |
| Toiminto | ● | | ● | | | ● |
| JÄRJESTELMÄT | | | | | | |
| Informaatio | ● | ● | ● | ● | ● | ● |
| Tosiaika | ● | | ● | ● | | ● |
| SUUNNITTELU-TASO | | | | | | |
| Järjestelmä | | | | | ● | |
| Systeemi | ● | ● | ● | ● | | ● |
| MENETELMÄ-TYYPPI | | | | | | |
| Analyysi | ● | ● | ● | | ● | |
| Suunnittelu | | ● | | ● | ● | ● |

Lähde: McClure, PC Tech Journal, 8/88 s. 61

[1] McClure, PC Tech Journal, 8/88 s. 57

TAS-välinettä hankittaessa eräs tärkeimmistä huomioon otettavista asioista on systeemityömenetelmä, jota sillä aiotaan tukea. Nolanin mallin avulla voidaan arvioida yrityksen tietojenkäsittelyn kehitysvaihe ja sen perusteella voidaan päätellä onko järkevä käyttää toiminto- vai tietokeskeistä systeemityömenetelmää. Burnsin ja Dennisin kontingenssimallin avulla voidaan toisaalta arvioida kehitettävien sovelluksien tyyppi ja sen perusteella valita sopiva lähestymistapa yrityksen sovelluskehitykseen: perinteinen vaihejako vai protoilu. Mallien käyttö ei ole toisiaan poissulkevia vaan niitä voidaan käyttää myös yhdessä arvioitaessa sopivaa systeemityömenetelmää yritykselle. Sen jälkeen kun on päätetty minkä tyyppinen systeemityömenetelmä on sopivin yritykselle voidaan vasta valita hankittava TAS-väline (taulukko 3).

Taulukko 3. ERÄIDEN TAS-VÄLINEIDEN TUKEMAT SYSTEEMITYÖMENETELMÄT JA MENETELMÄTYYPIT

| | Kuvausmenetelmä | Menetelmätyyppi |
|---|---|---|
| A/D TOOLKIT | Yourdon | Toimintokeskeinen systeeminsuunnittelu (SE) |
| CASE2000 | Yourdon<br>Käsiteanalyysi (Chen)<br>Ward/Mellor (real-time) | Toimintokeskeinen systeeminsuunnittelu (SE) (räätälöitävissä) |
| DEFT | Yourdon/Gane&Sarson<br>Jackson JSP<br>Käsiteanalyysi (useita) | Toimintokeskeinen systeeminsuunnittelu (SE) |
| IEF | Martin (IEM) | Tietosuunnittelu (IE) |
| IEF -- | Martin | Tietosuunnittelu (IE) |
| TEAMWORK | Yourdon<br>Käsiteanalyysi<br>Boeing (Real-time) | Toimintokeskeinen systeeminsuunnittelu (SE) |

Lähde: SYTYKE TAS-raportti s. 34

# 3. TIETOKONEAVUSTEINEN SYSTEEMITYÖ (TAS)

## 3.1 TAS:n historia

Tietokoneavusteinen systeemityö (TAS) alkoi kehittyä 1980-luvun alussa USA:ssa. Systeemityö oli ajautunut tilanteeseen, jossa toteutettavaksi päätettyjen tietojärjestelmien kehitysjättämä (backlog) oli vuosia. Vanhojen tietojärjestelmien ylläpitotyö vei suurimman osan suunnittelijoiden ja ohjelmoijien ajasta eikä aikaa näinollen jäänyt uusien tietojärjestelmien kehittämiseen. Ylläpitotyö vaati usein jopa 60-80 % koko ohjelmiston elinkaaren kustannuksista[1].

Systeemityön ongelmien ratkaiseminen, jotta se pystyisi tulevaisuudessa paremmin täyttämään muuttuvat tarpeet, vaatii Charetten mielestä:[2]

- Ohjelmistojen kasvavien kustannusten hillitsemistä
- Ohjelmistojen laadun ja ylläpidettävyyden parantamista
- Systeemityön tuottavuuden ja nopeuden parantamista

Tämä vaatii Charetten mielestä automatisoitujen systeemityömenetelmien käyttöönottoa.

Ensimmäiset TAS-välineet olivat 1980-luvun alussa markkinoille tuodut systeemityön dokumentointi- ja kaavio-ohjelmat. Ne olivat yksinkertaisia mikrotietokonepohjaisia piirto-ohjelmia. Kaavioilla ei ollut suoria yhteyksiä toisiin kaavioihin. Ne oli tarkoitettu tukemaan yksinomaan tiettyä kuvausmenetelmää,

---

[1] Charette, 1986 s. 11.

[2] Charette, 1987 s.4

kuten esimerkiksi tietovirta- tai kohdekaaviota.[1] Niistä puuttuivat virhetarkastukset eikä niillä katettu kuin rajoitettuja osia määrittely- tai suunnitteluvaiheista.

1980-luvun puolivälissä saatiin TAS-välineisiin kaksi tärkeää parannusta: virhetarkastukset ja kuvauskanta. TAS-välineellä tehtyjen kaavioiden oikeellisuus ja täydellisyys voitiin nyt tarkastaa ennen järjestelmän koodausta ja käyttöönottoa. Tarkastusten jälkeen kaaviot tallennettiin kuvauskantaan. Tämän jälkeen niitä oli helppo ylläpitää, jakaa projektin kesken ja käyttää myöhemmin uudelleen jossain muussa yhteydessä.[2]

Tällä hetkellä markkinoilla on TAS-välineitä, joilla voidaan kattaa lähes kaikki systeemityön vaiheet, esitutkimuksesta ylläpitoon. Ohjelmakoodi saadaan generoitua suunnitteluvaiheen määrityksistä. Näiden määritysten eheys ja täydellisyys voidaan tarkastaa ennen koodingenerointia. Tällä on huomattava vaikutus pyrittäessä loogisesti virheettömämpiin ohjelmiin. Nämä välineet ovat kuitenkin vielä täysin ATK-ammattilaisten TAS-välineitä, koska ne vaativat käyttäjältään hyvää systeemityön ja sen menetelmien tuntemusta.

## 3.2 TAS-välineen keskeiset ominaisuudet

TAS-välineen käytön kannalta keskeisin ominaisuus on kuvauskanta, jonka tulisi tukea käytettävää systeemityömenetelmää. Lisäksi automatisoitu systeemien

---

[1] McClure, 1989 s. 8.

[2] McClure, 1989 s.13.

kehitystyö vaatii TAS-välineeltä tehokkaita mallintamis- ja dokumentointiominaisuuksia.[1]

### 3.2.1. Kuvauskannan ominaisuudet

Tärkein apuväline automatisoidussa systeemityössä on siis keskustietovarasto, kuvauskanta. Kuvauskanta voi sijaita joko keskuskoneella tai henkilökohtaisella työasemalla. Tämä riippuu käytettävästä TAS-ympäristöstä. Kuvauskannat voidaan jakaa käyttäjälleen antamansa systeemityötuen perusteella neljään eri luokkaan[2]:

1. **Passiivinen tietohakemisto**
2. **Aktiivinen tietohakemisto**
3. **Kuvauskanta ATK-ammattilaiselle**
4. **Tietämystekniikkaa hyödyntävä kuvauskanta**

TAS-väline, joka käyttää passiivista tietohakemistoa, ei tarjoa suurta apua systeemityöhön. Siitä on apua tietokantojen toteuttamisessa sen jälkeen, kun tietotarpeet on määritelty ja suunnittelu on valmis. Nykyiset TAS-välineet eivät yleensä enää käytäkään tällaisia hakemistoja, vaan kehittyneempiä tietovarastoja. Passiivisia tietohakemistoja käytetään nykyään tyypillisesti 3. sukupolven ohjelmointikielten kanssa.

TAS-väline, joka käyttää aktiivista tietohakemistoa, antaa tukea tietokantojen suunnitteluun ja toteutukseen. Niillä voidaan tuottaa tietokantamäärityksiä suoraan kuvauksista. Vaikkakin ne antavat tukea tietokantojen kehitystyöhön, niin muuhun systeemityöhön niiden antama tuki on vähäistä. Ne ovat täysin riippuvaisia käyttäjänsä suunnittelutaidos-

---

[1] Finkelstein, 1987 s.52.

[2] Finkelstein 1987, s. 52-54

ta, joten ne eivät sovellu itsenäiskäyttäjien työkaluiksi. Nämä hakemistot ovat yleensä integroituja 4. neljännen sukupolven ohjelmointikielten kanssa.

Finkelsteinin esittämä kuvauskantojen luokitus passiivisiin ja aktiivisiin tietohakemistoihin on TAS-välineyhteydessä turha. Välineet, jotka käyttävät passiivisia hakemistoja eivät ole mielestäni TAS-välineitä, koska niiden hakemistot sisältävät vain sovellusten tietomäärityksiä eikä niihin pysty tallentamaan tietoja systeemien kuvauksista ja niiden välisistä yhteyksistä.

"ATK-ammattilaisten" kuvauskantaa käyttävät TAS-välineet tukevat vuorovaikutteisen graafisen käyttöliittymänsä avulla systeemityön analyysi- ja suunnitteluvaiheita. Ne eivät kuitenkaan anna käyttäjälleen apua itse suunnittelumenetelmän käytössä, joten ne ovat kokeneiden suunnittelijoiden, ATK-ammattilasten apuvälineitä. Graafisen käyttöliittymänsä avulla ne auttavat suunnittelijoita ja käyttäjiä kommunikoimaan keskenään. Suurimmassa osassa nykyisistä TAS-välineistä on tällainen kuvauskanta, jonne suunnittelutiedot tallennetaan.

Kaikkein kehittyneimmät kuvauskannat hyödyntävät tietämystekniikkaa (Artificial Intelligence, AI). TAS-välineet, jotka käyttävät tällaista hakemistoa, soveltuvat myös loppukäyttäjien työvälineiksi. Käyttäjän ei välttämättä tarvitse hallita itse systeemityömenetelmää, koska kuvauskanta opastaa häntä menetelmän oikeaoppisessa käytössä.

#### 3.2.1.1 Kuvauskannan tekniset ominaisuudet

TAS-välineen helppokäyttöisyys on merkittävä tekijä sekä systeemityön nopeutumisen ja käyttäjänsä työtyytyväisyyden kannalta. Kuvauskannan tulisikin

sisältää sen sujuvan käytön kannalta seuraavat tekniset ominaisuudet:[1]

1. **Suora yhteys kuvauksista kuvauskantaan ja sen sisältämiin tietomäärityksiin.** Systeemityön nopeuden ja siten tuottavuuden kannalta on tärkeää, että kuvaustilasta on suora yhteys kuvauskantaan. Toisissa välineissä joudutaan poistumaan kuvaustilasta, ennenkuin voidaan päivittää kuvauskannan tietomäärityksiä. Tämä hidastaa kaavioiden tekemistä ja turhauttaa niiden tekijää.

2. **Kuvauskohtaiset tiedostot kuvauskannassa.** Jokainen kuvaus tulisi olla omana tiedostonaan kuvauskannassa. Tämä helpottaa huomattavasti niiden ylläpitotyötä, koska niitä voidaan kopioida, siirtää, muuttaa, jne. erillisinä koko kuvauskannasta. Näihin erillisiin kuvaustiedostoihin tulisi olla suora yhteys välineestä, ilman että siitä täytyy ensin poistua. TAS-väline, joka tekee kaikki kuvaukset samaan tiedostoon on tässä mielessä hankalampi käyttää.

3. **Kuvauskannan tietojen suora päivitys.** TAS-välineen sujuvan käytön kannalta on myös tärkeää, että kuvauskanta mahdollistaa määritystensä suoran uudelleen nimeämisen. Toisissa välineissä haluttaessa esimerkiksi nimetä tietovirta uudelleen, täytyy 'vanha' tietovirta ensiksi poistaa ennenkuin 'uuden niminen' tietovirta voidaan tehdä.

4. **Kuvauskannan sisällön raportit.** Kuvauskannan tulee raportoida käyttäjälleen mihin kaikkiin kuvauksiin tietomääritysten tuhoaminen vaikuttaa. Tämä nopeuttaa kaavioiden ylläpitotyötä, koska käyttäjä saa heti tietoonsa mitkä kuvauk-

---

[1] Baram & Steinberg, 1989, s. 75-76

set tulee tarkastaa, kun hän tekee yhteen kuvaukseen muutoksia. Toisaalta tämä varmistaa, ettei käyttäjä vahingossa tuhoa jostain toisesta kuvauksesta tietoja, joita siinä tarvitaan. Näiden raporttien täydellisyys vaihtelee suuresti eri välineiden kesken.

5. **Tietojen vaihto-ominaisuudet.** Eri systeemityöprojektit saattavat käyttää samoja tietoja suunnittelutyössään. Onkin tärkeää, että kuvauskannoista saadaan tietoja eri projekteille. Kuvauskannat tulee olla joko verkossa tai sitten niissä täytyy olla tehokas tiedon vaihto-ominaisuus (export/import).

Eri TAS-välineissä on lukuisa määrä erilaisia teknisiä ominaisuuksia ja hienouksia. Ne eivät ole kuitenkaan niin tärkeitä ominaisuuksia kuin edellä mainitut. TAS-välinettä hankittaessa ei saisi antaa näille teknisille hienouksille liikaa painoarvoa, vaan valinnan tulisi keskittyä olennaisimpiin ominaisuuksiin.[1]

### 3.2.1.2 Kuvauskannan sisältämät tarkistukset

Kaikista järjestelmien elinkaaren aikana löydetyistä virheistä jopa yli 2/3 johtuu puutteellisesta tai väärästä määrittelystä ja suunnittelusta. Näistä virheistä löydetään kuitenkin vain 1/3 ennen järjestelmien käyttöönottoa. Myöhäisemmässä vaiheessa löydetyt virheet ovat kuitenkin paljon kalliimpia korjata kuin aikaisemmin löydetyt virheet.[2] Virhetarkastukset ovatkin TAS-järjestelmän olennainen osa. Automatisoitujen virhetarkastusten avulla mahdolliset virheet löydetään jo aikaisemmassa systeemityön

---

[1] Baram & Steinberg, 1989 s. 74

[2] Boehm 1981, s. 18

vaiheessa kuin ennen. Se, että virheet löydetään jo suunnitteluvaiheessa, parantaa puolestaan järjestelmien laatua ja vähentää tarvittavaa ylläpitotyötä. Automaattiset virhetarkastukset voidaan jaotella viiteen perustyyppiin:[1]

1. **Syntaksi- ja tyyppitarkastukset**
2. **Täydellisyystarkastukset**
3. **Toiminnallisten jaotteluiden tarkastukset**
4. **Ristiintarkastukset**
5. **Audit-trail -tarkastukset**

**1. Syntaksi- ja tyyppitarkastukset**

Syntaksitarkastuksella tarkastetaan, että järjestelmää kuvaavat kaaviot ovat menetelmän sääntöjen mukaisesti piirrettyjä. Esimerkiksi tietovirtakaaviosta tarkastetaan näin, että jokaiseen kuvattuun prosessiin tulee vähintään yksi tietovirta ja ettei kaksi tiedostoa ole suoraan kytketty toisiinsa. Tyyppitarkastuksella puolestaan tarkastetaan, että kuvaus on tehty loogisesti oikein. Esimerkiksi tietovirtakaaviossa merkitty prosessi on todella prosessi eikä esimerkiksi tietovirta. TAS-järjestelmää käytettäessä syntaksi- ja tyyppitarkastukset tapahtuvat yleensä samanaikaisesti kuvauksia tehtäessä. Näin pyritään estämään käyttäjää syöttämästä väärää tai epäloogista tietoa kuvauksiin. Tyyppitarkastukset vaatisivat toimiakseen tietämystekniikan hyväksikäyttöä. Tällainen TAS-väline pystyisi erottamaan esimerkiksi prosessit ja tietovirrat toisistaan. Tällaista ominaisuutta ei kuitenkaan vielä ole markkinoilla olevissa TAS-välineissä.

**2. Täydellisyystarkastukset**

Täydellisyystarkastuksella tarkastetaan, että kuvaus sisältää kaikki tarpeelliset tiedot. Tietovirtakaaviosta tarkastetaan esimerkiksi, että kaikki tieto-

---

[1] McClure 1989, 41-43.

virrat ovat nimetty ja jokaiseen prosessiin tulee ja siitä lähtee ainakin yksi tietovirta. Tällaisen yksinkertaisen, pinnallisen, tarkastuksen lisäksi tulisi varmistua siitä, että kaaviossa esiintyvistä kuvauksista on täydelliset määrittelyt kuvauskannassa ja että ne ovat eheitä muiden kaavioden määrittelyiden kanssa. Tällaiset tarkastukset suoritetaan yleensä tallennettaessa valmiita kaavioita kuvauskantaan.

**3. Toiminnallisten jaotteluiden tarkastus**

Toiminnallisten jaotteluiden tarkastuksella tarkastetaan hierarkkisista rakennekaaviosta (Hierarchical tree structure diagrams), että toiminnot ovat jaoteltu käytettävän kuvausmenetelmän sääntöjen mukaan alitoimintoihinsa. Siirryttäessä hierarkkiassa alaspäin, tulee tarkasteltavan tason toimintojen tarkentaa hierarkkiassa ylempänä olevaa "emotoimintoaan". Toisaalta rakennekaaviossa ei toiminto saa kutsua itseään[1]. Jokaisen alitoiminnon tulee myös sisältää yksityiskohtaisempaa tietoa toiminnosta kuin sen emotoiminto. Tämän tarkastusta kutsutaan semanttisen jalostumisen tarkastamiseksi.

Syntaksi- ja tyyppitarkastukset, täydellisyys- ja eheystarkastukset sekä toiminnallisten jaotteluiden tarkastukset ovat menetelmäsidonnaisia tarkastuksia. Nämä yksinkertaiset tarkistukset huolehtivat yksittäisen kuvauksen oikeellisuudesta. Koko kuvattavan järjestelmän oikeellisuudesta huolehtivat ristiin- ja audit-trail -tarkistukset.

**4. Ristiintarkastukset**

Rakenteiset kuvausmentelmät tukevat asteittain tarkentuvaa lähestymistapaa (top-down approach). Kuvaus aloitetaan järjestelmän yleisnäkemyksistä ja työn edetessä siirrytään yhä tarkemmalle tasolle.

---

[1] Martin/McClure 1988, 45-65.

Jokainen taso järjestelmästä kuvataan kaaviolla tai useilla kaavioilla ja kaikki kuvaustasot ovat yhteydessä toisiinsa, jolloin kuvauksissa voidaan siirtyä tarkemmalle tasolle niin haluttaessa. Esimerkiksi ylemmän tason tietovirtakaavion prosessista voidaan siirtyä tarkastelemaan ko. prosessin tietovirtoja. Ristiintarkastuksessa tarkastetaan tällaisten kerrostuneiden kuvausyhtymien eheys eri kerrosten välillä. Ristiintarkastuksella pyritään poistamaan epäloogisuudet ja virheellisyydet eri kuvausten ja kuvaustasojen väliltä. Tämä on eräs automaattisen koodigeneroinnin perusedellytyksistä. Ristiintarkastuksesta käytetään myös nimitystä tasapainoanalyysi ja se helpottaa löytämään yleensä vaikeasti havaittavat virheet järjestelmästä jo kehitysvaiheessa.

**5. Audit-trail -tarkastus**

Audit-trail -tarkastuksella pystytään analysoimaan vastaako tuotettu järjestelmä sille määrittely- ja suunnitteluvaiheessa asetettuja vaatimuksia. Esimerkiksi koodigeneraattorilla tuotetusta ohjelmakoodista voidaan uudelleen johtaa suunnitteluvaiheen kuvaukset, joita verrataan alkuperäisiin kuvauksiin. Audit-trail -tarkastus auttaa löytämään järjestelmän toteuttamattomat, mutta kuitenkin määritellyt, toiminnot sekä käyttämättömän syöttötiedon ja ohjelmakoodin.

TAS-väline raportoi virhetarkastuksien tuloksista joko kuvauksia tehtäessä tai niistä voidaan pyytää raportti paperille jälkeenpäin. Tarkastuksilla säästetään työtä etenkin ylläpitovaiheessa muutettaessa olemassa olevaa järjestelmää, koska kuvauskannasta saadaan tieto mihin kaikkiin kaavioihin tietty muutos vaikuttaa. TAS-välineen tehokkaan käytön kannalta sen tulisi pystyä tekemään vähintään täydellisyys- ja ristiintarkastukset[1].

---

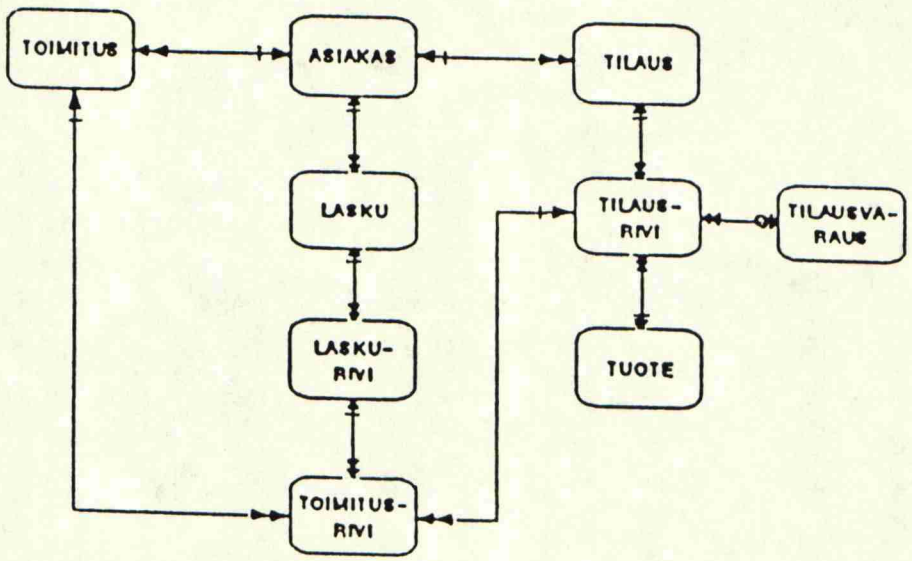
[1] Baram & Steinberg, 1989 s. 76

### 3.2.2 TAS-välineen mallintamisominaisuudet

TAS-välineen tarjoamien mallintamisominaisuuksien tulee Finkelsteinin[1] mielestä kattaa sekä tietojen että toimintojen mallintamisen.

Tietojen mallintamiseen käytetään kohdemallia. Kohdemalli koostuu kohdekaaviosta (Kuva 8.) ja tietomäärityksistä. Kohdekaaviossa ovat kohteet ja niiden väliset yhteydet graafisessa muodossa. Tietomääritykset kuvaavat kohdekaavion kohteiden staattisia ominaisuuksia, attribuutteja. Kohdemalli auttaa käyttäjiä täsmentämään tietotarpeensa.

Kuva 8. ESIMERKKI KOHDEKAAVIOSTA

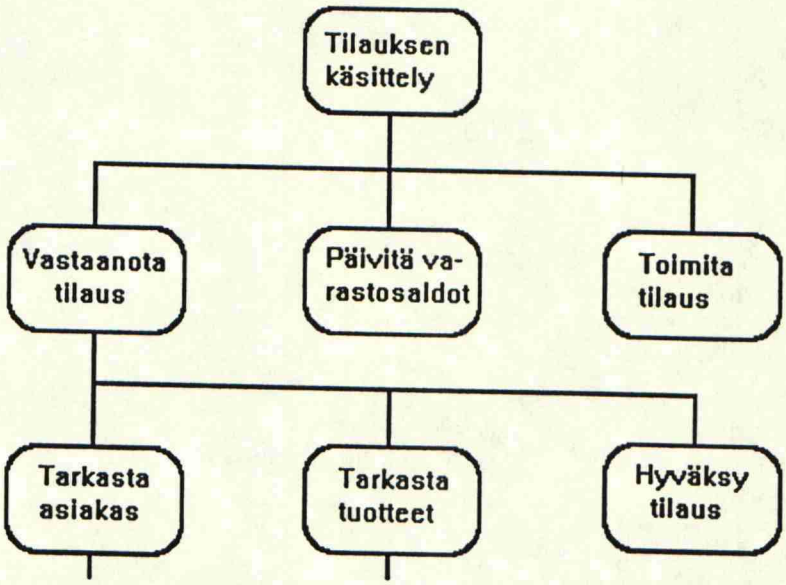



Toimintojen mallintamiseen käytetään puolestaan toimintomallia, joka koostuu toimintokaaviosta (Kuva 9.) ja siinä olevien toimintojen määrityksis-

[1] Finkelstein, 1988 s. 56

tä. Toimintomallien laatimiseen on useita eri tekniikoita. Näitä ovat mm. tietovirtakaaviot (data flow diagram), ohjelmarakennekaaviot (structure chart), toimintokaaviot (action diagram). Toimintomalli määrittelee tarvittavan logiikan, jolla kohdemallin tiedot käsitellään. Toimintokaavio tarjoaa graafisen esityksen toiminnoittain logiikasta ja yritystoimintatilanteista, joissa toiminnot suoritetaan. Toimintomalli tarjoaa käyttäjille tukea tarvitsemiensa toimintojen yksilöimisessä.

Kuva 9. ESIMERKKI TOIMINTOKAAVIOSTA




Tietomalli ja sen toimintomalli integroituvat kuvauskannan välityksellä. Samalla kuvauskanta huolehtii, että molempien mallien keskinäinen eheys säilyy, vaikka toista muutettaisiinkin.

TAS-välineet tarjoavat joko passiivista tai tekoälypohjaista mallintamisen tukea systeemityöhön. TAS-väline, joka tarjoaa vain passiivista mallintamistukea, ei pysty neuvomaan käyttäjäänsä menetelmänsä

käytössä. Ne tarjoavat vain luonnosteluominaisuuksia graafisten tieto- ja toimintokuvauksien tekoon ja ylläpitoon. Kaaviot ja niiden määrittelyt sekä kuvaruutumääritykset pitää manuaalisesti syöttää koneelle. Myös näiden ylläpitotyö vaatii manuaalista työstöä koneella. Automatisoitua tukea voidaan hyödyntää vain kaavioiden fyysisessä työstöprosessissa sekä osittain niiden eheystarkastuksissa.

Tällaiset passiiviset suunnitteluapuvälineet eivät kykene saamaan itsenäiskäyttäjältä riittävästi suunnittelutietoja, jotta se pystyisi niistä generoimaan ohjelmakoodia. Vaikkakin ne tarjoavat näyttävää graafista tukea suunniteluun, on niiden antama hyöty hyvin pinnallista. Ne ovat täysin riippuvaisia käyttäjänsä suunnittelutaidoista, joten ne eivät sovellu itsenäiskäyttäjälle. Näiden välineiden kyky parantaa systeemityön tuottavuutta on erittäin rajallinen, johtuen niiden passiivisesta käyttäjäkytkennästä.

Suurin osa tämän hetkisistä TAS-välineistä tarjoavat ainoastaan passiivisia mallintamisominaisuuksia. Eheystarkastukset kaavioiden ja näiden määritysten täytyy tehdä tämän vuoksi manuaalisesti. Tämä puolestaan lisää virhemahdollisuuksien määrää. Virhetarkastuksia kaavioden välillä pystytään osittain tekemään jo parhailla tämän hetkisillä välineillä, mutta näiden tarkastustulosten tulkitseminen on vielä itsenäiskäyttäjille liian vaativaa.

Tekoälypohjaisia mallintamisominaisuuksia sisältävät TAS-välineet puolestaan tekevät suuren osan kaavioiden piirrostyöstä sisäänrakennetun tietämysjärjestelmänsä avulla. Itsenäiskäyttäjät syöttävät tietomäärittelynsä, jotka kuvauskanta heti tarkastaa ja hyväksyy. Näistä määrittelyistä kuvauskanta muodostaa automaattisesti graafisen tietomallin, kohdekaavion. TAS-väline analysoi tiedot ja tekee niistä

automaattisesti kohderyhmiä. Kohderyhmä sisältää ne kohteet, jotka voidaan toteuttaa erillisinä muista kohteista. Se ryhmittelee myös kohderyhmiä yhteen, mahdollisesti toteutettaviksi sovellusalueiksi (implementation clusters).

Toteutettavat kohderyhmät esittävät mahdollisia sovelluksia ja niiden kattamia liiketoimintoja. Automaattisesti muodostetut kohderyhmät auttavat johtoa asettamaan toteutettavat sovellukset järkevään toteutusjärjestykseen.

Tällaiset TAS-välineet johtavat tekoälynsä avulla automaattisesti tietomallista ja sen määrityksistä toimintomallin rungon ja peruslogiikan, joka tarvitaan määritetyn tiedon käsittelyyn. Itsenäiskäyttäjä voi tehdä TAS-välineen tekoälyn avustamana siihen tarvittaessa tarkennuksia ja muutoksia. Tarkennukset päivittyvät automaattisesti kuvauskantaan.

### 3.2.3 TAS-välineen dokumentointiominaisuudet

Systeemikuvauksia tuotetaan paljon läpi koko systeemin elinkaaren. Niitä täytyy myös runsaasti ylläpitää. Dokumentoinnin suurin ongelma on sen oikeellisuuden säilyttäminen. Kiireisessä systeemityössä tingitään yleensä ensimmäisenä dokumentointiin käytettävästä ajasta. Tämän vuoksi niiden luotettavuus saattaa kärsiä. Kuvauskannan onkin siis tärkeä automatisoida myös dokumentointia. TAS-väline voi tukea joko manuaalista dokumentointia tai se voi käyttää hyväkseen tietämystekniikka, jolloin dokumentit muodostuvat automaatisesti systeemikuvauksista.

Manuaalista dokumentointia tukeva väline tarjoaa tietokoneellisessa muodossa olevan varaston graafisten ja tekstimuotoisten dokumenttien tuottamiseen

ja ylläpitoon. Tämä dokumentaatio täytyy manuaalisesti syöttää ja määritellä koneelle. Jokainen muutos kuvauksiin täytyy uudelleen määritellä dokumentteihin ja syöttää kuvauskantaan. Dokumenttien ja suunnittelutulosten eheystarkastukseen saadaan vain vähäistä apua. Dokumenttien laatu riippuu täysin systeemityöntekijöiden ammattitaidosta.

Kehittyneemmän TAS-välineen tietämyspohjainen dokumentointi päivittää suunnittelumuutokset automaattisesti myös systeemidokumentteihin. Manuaalista dokumentointia ei tarvita.

Finkelsteinin esittämä TAS-välineiden jako manuaalista tai tietämyspohjaista dokumentointia tukeviin välineisiin perustuu mielestäni ns. perinteiseen systeemityöhön, jossa määrittely- ja suunnittelukuvauksista toteutetaan manuaalisesti ohjelmat. Kuvauksilla ja ohjelmilla ei siis ole fyysistä yhteyttä, jolloin on erittäin tärkeää, että systeemidokumentit ovat kunnossa. Jos kuitenkin käytetään tietokoneavusteista systeemityötä perinteisen systeemityön sijasta, ratkeaa dokumentointiongelma mielestäni automaattisesti. Ohjelmat generoidaan suoraan kuvauksista ja määrityksistä. Jos ohjelmaan tehdään muutoksia, ne tehdään kuvauksiin ja määrityksiin eikä ohjelmakoodiin. Tämän jälkeen ohjelmakoodi generoidaan uudelleen. Ohjelmat ovat siis itse dokumenttejaan eikä ole enää järkevää puhua erillisestä dokumentoinnista tai TAS-välineen tarjoamasta dokumentoinnin tuesta. TAS-välineitä käytettäessä systeemidokumentit ovat vain eri esitysmuodossa johon perinteisesti on totuttu.

### 3.2.4 TAS-välineiden tulevaisuuden ominaisuuksia

TAS-välineisiin on tulossa parannuksia 90-luvulla. Ehkä merkittävin parannus tulee olemaan tietämys-

tekniikan soveltaminen. Tämä tulee olemaan huomattava edistysaskel TAS-välineissä. Tällä hetkellä TAS-välineet ovat täysin ATK-ammattilaisten välineitä, mutta tietämystekniikan avulla niistä saadaan apuväline myös itsenäiskäyttäjille. TAS-välineisiin tarvitaan tietämystä kolmelta eri alueelta: Liiketoiminnasta, systeemityöstä ja sen menetelmistä sekä itse TAS-välineestä. Liiketoimintatietämys ohjaa käyttäjää suunnittelemaan sitä tukevia järjestelmiä. Systeemityötietämys puolestaan huolehtii käytettävän menetelmän oikeaoppisesta käytöstä ja TAS-välinetietämys ohjaa itse välineen fyysisessä käytössä. Systeemityötietämys mahdollistaa myös mm. tyyppitarkastuksen.

Toinen merkittävä parannus tulee olemaan vastakkaissuunnittelu (reverse engineering). Vastakkaissuunnittelulla tarkoitetaan prosessia, jossa jo valmiista ohjelmakoodista tuotetaan analyysi- ja suunnitteluvaiheen kuvaukset, jotka tallennetaan kuvauskantaan. Tämän jälkeen kuvauksiin tehdään tarvittavat muutokset ja ohjelmakoodi generoidaan uudelleen.[1] Tällä helpotetaan huomattavasti ohjelmien ylläpitotyötä. Ohjelmien muutokset ja korjaukset tehdään paljon korkeammalla abstraktiotasolla kuin ennen. Vastakkaissuunnittelulla arvioidaan olevan huomattava vaikutus ohjelmistotuotannon kehitysjättämän (backlog) purkamisessa. Markkinoilla on valtava määrä vanhoja III-sukupolven ohjelmointikielillä (esim. COBOL:lla) tehtyjä ohjelmia, joiden ylläpitotyöhön kuluu tällä hetkellä suuri osa yritysten tietojenkäsittelyresursseista. Tehokkaiden vastakkaissuunnitteluvälineiden avulla pystytään vanhojen, ei-TAS-välineillä alunperin tehtyjen, ohjelmien ylläpitotyö tekemään nopeammin ja tehokkaammin. Tällöin jää myös ohjelmien uustuotantoon ja kehitystyöhön enemmän aikaa.

---

[1] Margolis, 1988

Kolmas merkittävä muutos TAS-välineissä tulee olemaan niiden sovellusalueen mukainen eriytyminen. Markkinoille tullee TAS-välineitä, jotka ovat tarkoitettuja pelkästään tietyn sovellusalueen systeemityöhön, kuten taloushallinto, prosessitekniikka, tehdasautomaatio, jne. Näitä TAS-välineitä pystytään myös huomattavasti räätälöitämään ja laajentamaan aivan kuten nykyisiä valmisohjelmistopakettejakin.[1] TAS-välineiden sovellusalueen mukainen eriytymistarve johtunee osaltaan siitä, että niissä hyödynnetään tulevaisuudessa yhä enemmän tietämystekniikkaa. Yhteen TAS-välineeseen voidaan kuitenkin upottaa tekoälyä vain hyvin rajatulta sovellusalueelta ilman että näiden välineiden käyttönopeus hidastuisi liikaa ja ettei niiden koko kasvaisi liiaksi.

## 3.3 TAS-välineiden luokitteluja

### 3.3.1 Finkelsteinin luokittelu

TAS-välineet voidaan Finkelsteinin mielestä luokitella kolmeen pääryhmään niiden tarjoaman kuvauskannan sekä mallintamis- ja dokumentointiominaisuuksien perusteella:[2]

1. Tietokoneavusteiset suunnitteluvälineet
2. ATK-ammattilaisen TAS-välineet
3. Itsenäiskäyttäjän TAS-välineet

**1. Tietokoneavusteiset suunnitteluvälineet**

Nämä välineet tarjoavat automatisoitua tukea joko systeemityön analyysi-, suunnittelu- tai toteutusvaiheisiin rakenteisen analyysi- (SA, Structured

---

[1] CASE OUTLOOK, vol. 2, No. 3, 1988, pp. 1-7.
CASE OUTLOOK, vol. 2, No. 4, 1988, pp. 1-17

[2] Finkelstein, 1987 s. 52

Analysis) ja suunnittelunmenetelmän (SD, Structured Design) pohjalta. Välineet tukevat monia eri variaatioita näistä toimintokeskeisistä menetelmistä, kuten Yourdon, De Marco ja Gane - Sarson. Joillekin tietokeskeisille menetelmillekin, kuten Jackson ja Warnier-Orr, löytyy näistä välineistä tukea. Tällaisilla välineillä tuotetaan lähinnä yksittäisiä kaavioita, jotka eivät välttämättä ole yhteydessä muihin saman järjestelmän kuvauksiin.

Suurin osa näistä tietokoneavusteisista suunnitteluvälineistä käyttävät kuvauskantaa, joka ei hyödynnä tietämystekniikkaa. Järjestelmien mallintaminen ja dokumentointi tällaisilla välineillä on riippuvainen käyttäjänsä suunnittelutaidoista. Ne vaativat käyttäjältään erittäin hyvää tuntemusta tukemastaan systeemityömenetelmästään. Koska näissä välineissä ei ole virhetarkastuksia, ovat kuvauksien ja määrittelyiden täydellisyys, täsmällisyys ja eheys täysin käyttäjänsä vastuulla. Niinpä ne eivät sovikaan itsenäiskäyttäjien TAS-välineiksi. Voimakkaan ATK-ammattilaiskeskeisyytensä johdosta ne eivät sovellu käytettäväksi kuin kahdessa ensimmäisessä Nolanin tietojenkäsittelyn kehitysvaiheessa (taulukko 4.).

**2. ATK-ammattilaisen TAS-välineet**

Nämä välineet tarjoavat yleensä tietokoneavusteisten suunnitteluvälineiden ominaisuuksien lisäksi lisätukea tiedon ja toimintojen mallintamiseen. Jotkut näistä välineistä tukevat toimintokeskeisten menetelmien (SA/SD) sijasta tietokeskeistä menetelmää, kuten esimerkiksi Information Engineering (IE). Tällä hetkellä, kun puhutaan TAS-välineistä, tarkoitetaan yleensä näitä ATK-ammattilaisten TAS-välineitä.

Suurin osa näistä ATK-ammattilaisten TAS-välineistä, kuten tietokoneavusteisista suunnitteluvälineistäkin, käyttävät kuvauskantaa, joka ei hyödynnä tietämystek-

niikkaa. Niinpä näilläkin välineillä tuotettujen kuvauksien laatu riippuu käyttäjänsä suunnittelutaidoista. Tieto- ja toimintomallit sekä rakennekaaviot syötetään manuaalisesti koneelle. Nämäkään välineet eivät pysty generoimaan kohde- ja toimintokaavioita suoraan käyttäjänsä määrityksistä. Kaaviot täytyy myös ylläpitää manuaalisesti.

Nämä ATK-ammattilaisten TAS-välineet tukevat, parannuksena tietokoneavusteisiin suunnitteluvälineisiin, tiettyjen sisäänrakennettujen säännöstöjen avulla automaattista dokumentointia. Näihin ATK-ammattilaisten TAS-välineisiin on myös lisätty virhetarkastuksia, joilla kuvauksien laatua voidaan valvoa.

Käyttäjien liittäminen suunnitteluprosessiin näidenkin TAS-välineiden avulla on passiivista. ATK-ammattilaista tarvitaan "välittäjäksi". ATK-painottuneisuudestaan johtuen nämä välineet soveltuvat käytettäviksi Nolanin tietojenkäsittelyn kehitysvaiheessa 1-3 (taulukko 4).

Tälläisia ATK-ammattilaisten TAS-välineitä ovat mm. Knowledgewaren IEW, Texas Instrumentin IEF, Index Technologyn Excelerator ja Nastecin CASE 2000. Nämä ohjelmistot hallitsevat tällä hetkellä hyvin suurta osaa TAS-välineiden markkinoista[1].

### 3. Itsenäiskäyttäjien TAS-välineet

Itsenäiskäyttäjien TAS-välineet tarjoavat ATK-ammatilaisten TAS-välineiden ominaisuuksiin paljon käyttäjäystävällisemmän käyttöliittymän. Itsenäiskäyttäjien TAS-välineissä hyödynnetään tehokkaasti tietämystekniikkaa. Ne tarjoavat älykkään kuvauskannan, joka pystyy ohjaamaan käyttäjäänsä järjestelmien mallinta-

---

[1] SYTYKE TAS-raportti, 1988 s. 23

misessa. Nämä TAS-välineet pystyvät generoimaan kohde-ja toimintokaaviot suoraan käyttäjien liiketoimitatason määrityksistä. Itsenäiskäyttäjän TAS-välineiden kuvauskanta sisältää myös tekoälypohjaisia dokumentointiominaisuuksia.

Itsenäiskäyttäjien TAS-välineissä on ATK-ammattilaisen tietämys ja myös tietämys menetelmästä, jota väline käyttää. Näitä TAS-välineitä voivat itsenäiskäyttäjät käyttää ilman ATK-ammattilaisen avustusta. Näin ne soveltuvat käytettäviksi kaikissa tietojenkäsittelyn kehitysvaiheissa (taulukko 4). Tällaisia TAS-välineitä saataneen kuitenkin markkinoille vasta 90-luvun puolivälin jälkeen.

Taulukko 4. TAS-VÄLINEIDEN SOVELTUVUUS ERI TIETOJENKÄSITTELYN KEHITYSVAIHEESSA

| TAS-välineluokka | Tietojenkäsittelyn kehitysvaihe | | | | | |
|---|---|---|---|---|---|---|
| | toiminto-ohjattu | | | tieto-ohjattu | | |
| | 1 | 2 | 3 | 4 | 5 | 6 |
| TA-analyysi/ohjelmointivälineet | ——— | | | | | |
| ATK-ammattilaisen TAS-välineet | ——— | ——— | ——— | | | |
| Itsenäiskäyttäjän TAS-välineet | ——— | ——— | ——— | ——— | ——— | ——— |

Lähde: Finkelstein, 1988 s. 59

### 3.3.2 Aclyn luokittelu

TAS-välineet voidaan luokitella myös sisältämiensä ominaisuuksien perusteella 1. ja 2. sukupolven välineisiin. Ensimmäisen sukupolven TAS-tuotteet tulivat

markkinoille 1983-84. Näille ensimmäisen sukupolven TAS-tuotteille on luonteenomaista:[1]

**1. Rakenteellinen kuvausmenetelmä (structured technique)**
TAS-välineen tarkoituksena on tukea pääasiassa analyysivaiheen toimintoja. Esitutkimus- ja suunnitteluvaiheen tuki on vähäisempää. Ohjelmat tukevat graafisilla ominaisuuksillaan 1970-luvulla suosituiksi tulleita rakenteellisia kuvausmenetelmiä.

**2. Tietokoneavusteisuus (computer assistance)**
Tietokoneistettua tukea saadaan aiemmin manuaalisisiin systeemityön toimintoihin, kuten esimerkiksi kaavioiden piirtämiseen.

**3. Top-Down lähestymistapa (Top-Down approach)**
Nämä 1. sukupolven TAS-välineet ovat ensimmäisiä tuotteita, jotka tukevat rakenteisten kuvausmenetelmien Top-Down lähestymistapaa sovelluskehityksessä.

**4. Kuvauskanta (Proprietary dictionary)**
Jokaisessa tuotteessa on sen keskeisenä kuvauskanta. Kuvauskannat ovat kuitenkaan standardoimattomia ja jokaisella toimittajalla on omansa.

**5. Kuvausten siirrettävyys (Conceptual portability)**
Kuvausten siirettävyydellä tarkoitetaan sitä, että ne ovat laite-, käyttöjärjestelmä-, tietokannan hallintajärjestelmä-, ja ohjelmointikieliriippumattomia.

Aclyn mielestä suurin hyöty ensimmäisen sukupolven TAS-välineistä saadaan analyysivaiheessa. Suurin ongelma puolestaan näissä 1. sukupolven TAS-välineissä on mielestäni kuvauskantojen yhteensopimattomuus. Tuotteet kattavat vain pienen osan systeemityön vaiheista eikä niitä voida kuvauskantojen yhteensopimattomuuksien

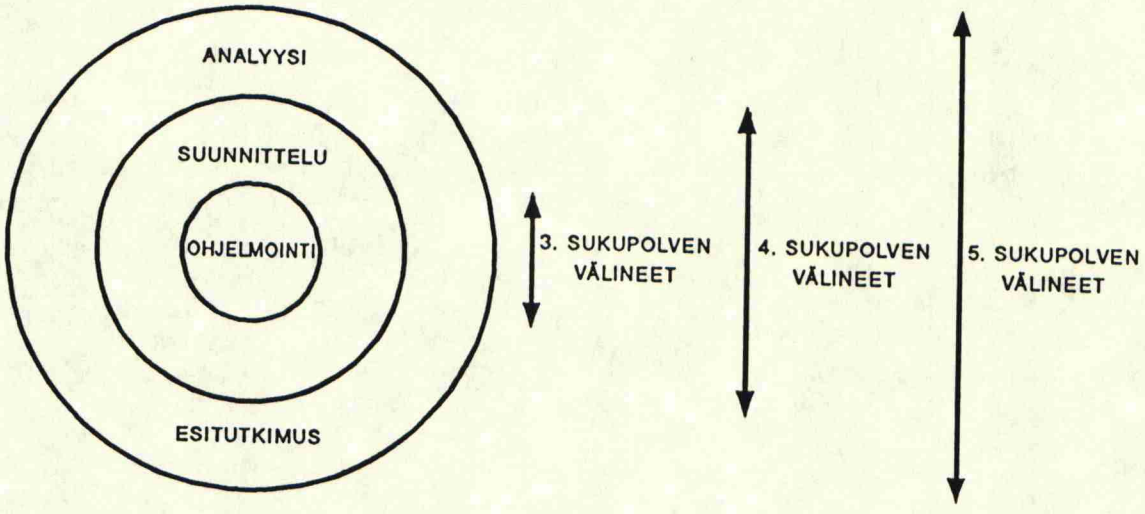
---

[1] Acly, 1987, s. 6.

vuoksi liittää keskenään kattavammaksi "ohjelmapaketiksi".

Toisen sukupolven TAS-välineet laajentavat Aclyn mielestä TAS(CASE-)-käsitettä niin, että hänen mielestään olisikin syytä puhua tietokoneavusteisesta systeemin kehitys- ja ylläpitoympäristöstä (CADME, Computer-Aided Development and Maintenance Environments). Tietokoneavusteisen systeeminkehitys- ja ylläpitoympäristö on Aclyn mielestä ensiaskel kohti viidennen sukupolven ohjelmointikieliä (5GL's).

Kuva 10. ERI SUKUPOLVEN OHJELMOINTIKIELTEN KATTAVUUS SYSTEEMITYÖN VAIHEISTA




Lähde: Acly, 1987 s. 7

Kolmannen sukupolven ohjelmointikielet kehitettiin helpottamaan ohjelmointia ja ohjelmakoodin ylläpitoa sekä parantamaan systeemityön tuottavuutta. Neljännen sukupolven ohjelmointikielet laajensivat tietokoneavusteisuutta kattamaan, ainakin osoittain, suunnitteluvaiheen. Sovelluksien osia, kuten esimerkiksi näytöt ja

tietokannat, saatiin generoitua suoraan suunnittelijan määrityksistä.

Viidennen sukupolven ohjelmointikielet laajentavat edelleen tietokoneavusteisuutta. Ne kattavat myös esitutkimuksen ja analyysivaiheen toimintoja. Tavoitteena on, että suurin osa sovelluksista saadaan generoitua suoraan analyysivaiheen jälkeen. Tähän päästään käyttämällä, vielä tosin suunnitteilla olevia, koodigeneraattoreita, jotka pystyvät tulkitsemaan analyysivaiheessa normaalin kielen terminologialla tehtyjä määrityksiä. Tällä hetkellä joudutaan analyysivaiheen suunnitelmat 'kääntämään' suunnitteluvaiheessa koodigeneraattoreiden ymmärtämälle terminologialle.

Kukin ohjelmointikielen sukupolvi lisää tietokoneavusteisuuden kattavuutta systeemityössä. Aclyn mielestä viides sukupolvi kattaa kaikki systeemityön vaiheet. Viidennen sukupolven ohjelmointikieliä voitaisiinkin tämän perusteella kutsua myös kokonaisvaltaiseksi TAS-ympäristöiksi, joita Acly nimittää toisen sukupolven TAS-välineiksi. Toisen sukupolven TAS-välineilläkin on omat keskeiset ominaisuudet ovat Aclyn mielestä:[1]

**1. Integroidut ympäristöt (Environments)**
Ensimmäisen sukupolven TAS-välineet ovat pääasiassa yksittäisiä, tiettyyn systeemityön vaiheeseen kehitettyjä tuotteita. Toisen sukupolven TAS-välineiden keskeisin ominaisuus on eri systeemityön vaiheiden integrointi keskenään, jolloin ylemmän tason määritykset saadaan suoraan alemman tason lähtötiedoiksi.

**2. Hajautettu sovelluskehitys (Distributed/Cooperative Application Development)**
TAS:tä tehdään yhä enemmän henkilökohtaisilla työ-

---

[1] Acly, 1987 s. 8

asemilla, jotka ovat tehokkaampia ja halvempia käyttää kuin keskuskoneet.

**3. Laajennettu ja jaettava kuvauskanta (Enhanced, Shared Dictionaries/Directories)**
Kuvauskannat laajennetaan tukemaan esitutkimusta ja analyysivaiheen toimintoja. Yrityksen metamallit (kohde- ja toimintomallit) tallennetaan keskuskuvauskantaan, joka yleensä sijaitsee keskuskoneella, josta se on jaettavissa (downloadable) eri projektien ja TAS-välineiden kesken.

**4. Tietämystekniikka (Artificial Intelligence)**
Toisen sukupolven TAS-välineet pystyvät kääntämään tietämystekniikkaa hyväksi käyttäen suunnittelijan tai jopa itsenäiskäyttäjän määrittelyt suoraan analyysivaiheesta toimivaksi ohjelmaksi. Tietämystekniikan avulla TAS-välineet voivat myös ohjata tottumatonta käyttäjää TAS-menetelmän käytössä ja tarkastaa hänen tekemiensä määritystensä oikeellisuuden.

Toisen sukupolven TAS-välineitä tullee markkinoille todennäköisesti 90-luvunpuolivälin jälkeen. Osa tämän hetkisistä TAS-välineistä on jo nyt hyvin lähellä Aclyn 2. sukupolven TAS-välinettä. Suurin puute niissä tällä hetkellä on tietämystekniikan hyödyntämättömyys.

### 3.3.3 Gibsonin luokittelu

TAS-välineet voidaan luokitella myös sen mukaan, mihin systeemityön vaiheeseen ne on tarkoitettu. Gibson luokitteleekin TAS-välineet:[1]

1. **Tietosuunnittelun välineet (Upper CASE)**
2. **Systeeminsuunnittelun välineet (Middle CASE)**
3. **Systeemintoteutuksen välineet (Lower CASE)**

---

[1] Gibson, BYTE, April 1989 s. 209

**1. Tietosuunnittelun välineet (Upper CASE)**

Tietosuunnittelun välineillä kuvataan yrityksen toimintaa ja suunnitelmia. Perustana näissä välineissä on yrityksen metatietomalli, jossa on kuvattuna kaikki yritystoimintaan liittyävt tiedot ja tietotarpeet. Lisäksi ne käyttävät graafisia hajautuskuvauksia (decomposition diagrams), joiden avulla määritellään ja analysoidaan yrityksen toimintaa. Yrityksen toiminnot (function) voidaan kuvata esimerkiksi liiketoimintayksikkö- tai osastotasolla . Lisäksi yrityksen tavoitteet ja kriittiset menestystekijät, kuten myös resurssit, vastuut ja nykytilan ongelmat voidaan kuvata graafisesti. Kuvaukset auttavat ymmärtämään paremmin yrityksen toimintaa.

Tällaisen, koko yrityksen kattavan, kuvauksen tekeminen on erittäin työläs. On kuitenkin huomattava, että tämän tason suunnitelmat ovat melko pitkäikäisiä. Lisäksi, käytettäessä tietokoneavusteista suunniteluvälinettä, niiden päivittäminen on helppoa. Yksi suurimmista eduista on kuitenkin se, että näitä kuvauksia voidaan myöhemmin käyttää hyväksi systeeminsuunnitteluvälineillä (middle CASE). Tämä toisaalta helpottaa systeeminsuunnittelutyötä ja toisaalta parantaa järjestelmien tietojen yhtenäisyyttä.

**2. Systeeminsuunnittelun välineet (Middle CASE)**

Systeeminsuunnittelun välineillä mallinnetaan yksittäisiä tietojärjestelmiä ja niiden toimintoja sekä tietotarpeita. Toivottavaa olisi, että yrityksellä olisi käytössään tietosuunnitteluväline, jonka kuvauksista saataisiin lähtötiedot systeeminsuunnittelutyölle.

Tällä hetkellä systeeminsuunnittelun välineet ovat eniten myytyjä TAS-välineitä.[1] Osin ehkä tästä johtuen

---

[1] SYTYKE TAS-raportti, 1988 s. 23

TAS:stä puhuttaessa yleensä tarkoitetaan tietokoneavusteista systeeminsuunnittelua. Tämä on kuitenkin vain osa kokonaisvaltaisesta TAS-ajattelusta. Näitä systeeminsuunnittelun välineitä kutsutaan myös Front-end-välineiksi[1].

**3. Systeemintoteutuksen välineet (Lower CASE)**

Systeemintoteutuksen välineillä tehdään systeeminsuunnittelun kuvauksiin teknisiä tarkennuksia, jotta koodigeneraattorit osaisivat tulkita niitä oikein. Tällaisia tarkennuksia ovat esimerkiksi erilaiset tietokantamääritykset. Näitä systeemintoteutuksen välineitä kutsutaan myös Back-end -välineiksi[2].

TAS-ideologia korostaa, että kaikkien tietojärjestelmien määrittelyjen, esitutkimuksesta toteutukseen ja ylläpitoon, tulee olla täysin integroituja. Käytettävän TAS-järjestelmän pitääkin tämän mukaan kattaa kaikki systeemityön vaiheet. Käytännössä ongelmia aiheuttaa se, ettei markkinoilla vielä ole TAS-tuotetta, joka kattaisi täysin kaikki vaiheet, eikä kuvauskantojen standardien puuttumisen vuoksi ei ole mahdollista koota erillisistä TAS-välineistä kaikki vaiheet kattavaa järjestelmää.

Ideaalitilanteessa organisaatiossa on käytössä täysin integroitu TAS-järjestelmä (integrated CASE, ICASE). Yrityssuunnittelijat tekevät liiketoiminnalliset määrittelyt, jotka määrittävät yritystoiminnan perusaktiviteetit, tietosunnitteluvälineillä (upper CASE). Nämä määrittelyt tallentuvat kuvauskantaan, josta ne voidaan hakea systeeminsuunnittelijoiden välineisiin (middle CASE) lähtötiedoiksi. Systeeminsuunnittelijat laajentavat ja tarkentavat näitä määrityksiä suunnitteilla olevan tietojärjestelmän alueelta. Tulokset tallentuvat jälleen kuvauskantaan, josta ne saadaan

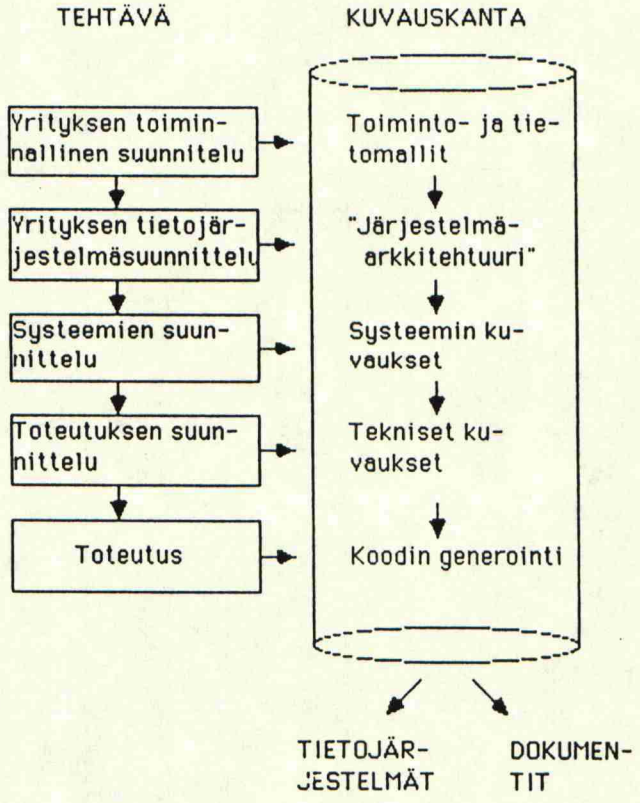
---

[1] CASE-Outlook, vol.2 no.3 1988

[2] CASE-Outlook, vol.2 no.3 1988

systeemin toteutusvaiheen välineille (lower CASE) lähtötiedoiksi, joihin tehdään tarvittavat tarkennukset varsinaista koodigenerointia varten (Kuva 11). Tällainen TAS-järjestelmä tukee hyvin Zachmanin tietojärjestelmäarkkitehtuurin näkemystä[1].

Kuva 11. TÄYSIN INTEGROITU TAS-JÄRJESTELMÄ




Tekijän mukaelma kuvasta BYTE, April 1989 s. 210

## 3.4 TAS:n vaikutus ohjelmistojen elinkaareen

Ohjelmiston elinkaari on monivaiheinen prosessi, joka alkaa ongelman määrittelystä ja jatkuu ohjelmiston ylläpitoon. Elinkaari ymmärretään yleensä viisivaiheiseksi prosessiksi, jonka vaiheet ovat (Kuva 12):[2]

---

[1] Zachman, 1987, s. 276-292.

[2] Burkhard & Jenster 1989, s. 28
McClure 1989, s. 184

1. Analyysi
2. Suunnittelu
3. Koodaus
4. Testaus ja käyttöönotto
5. Käyttö ja ylläpito

Analyysivaiheessa määritellään tietojenkäsittelytarpeen ratkaisevan ohjelmiston vaatimukset. Käyttäjien tarpeet kartoitetaan, jotta niitä tyydyttävä ratkaisu pystytään määrittelemään. Myös toteutettavan ohjelmiston rajoitukset ja suorituskyky sekä tarvittavat toiminnot määritetään. Analyysivaihe tuottaa ohjelmistomäärittelyn (system specification) tarkassa, formaalissa ja testattavassa muodossa. Tämä dokumentti on suunnitteluvaiheen lähtökohta. Analyysivaihe voidaan, ainakin osittain, korvata myös protoilemalla. Tällöin toteutettavasta tietojärjestelmästä tehdään prototyyppi, jonka avulla voidaan käyttäjien kanssa testata, että suunnittelijat ovat ymmärtäneet oikein käyttäjien tarpeet.[1]

Suunnitteluvaiheessa suunnitellaan, kuinka analyysivaiheessa määritelty ohjelmisto toteutetaan. Toimintojen logiikat kuvataan, kuten myös niiden tarvitsemat tiedot. Tämän jälkeen nämä suunnitteluvaiheen kuvaukset koodataan tietokoneen ymmärtämiksi komennoiksi (koodausvaihe).

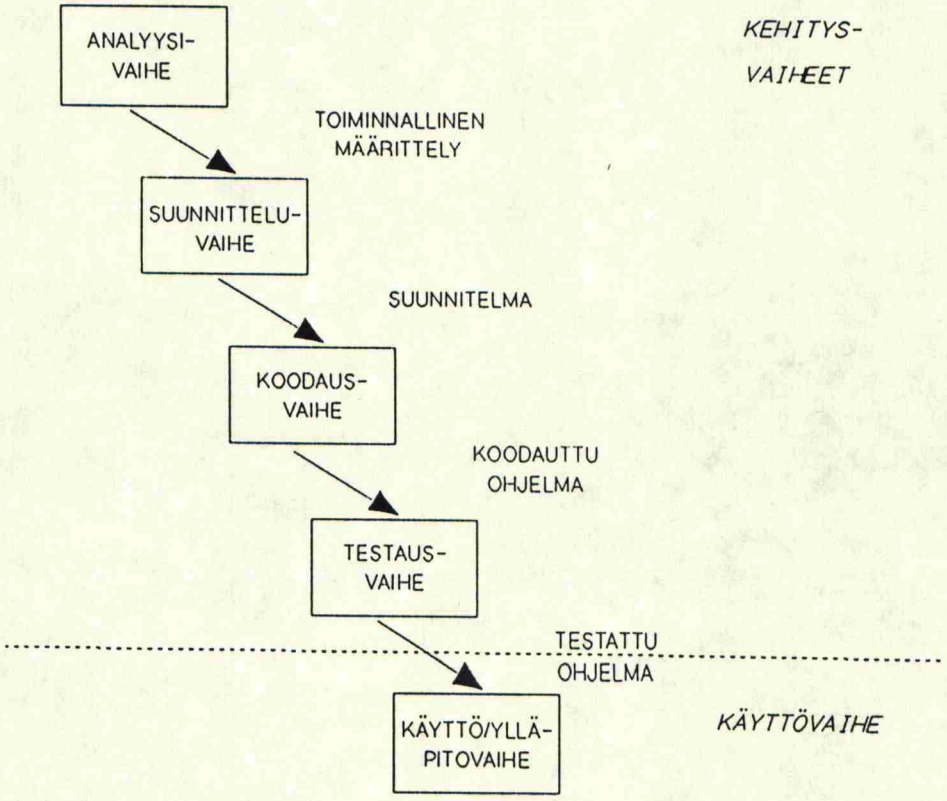
Testausvaiheessa kokeillaan täyttääkö toteutettu ohjelmisto sille asetetut vaatimukset ja käyttääkö se syöttötietojaan loogisesti oikein. Testaus suoritetaan yleensä ensin pieninä ohjelmiston osina (modulitestaus), jonka jälkeen osat yhdistetään ja testataan toimivatko ne yhdessä (järjestelmätestaus).

---

[1] Martin 1982, s. 64

Kun ohjelmisto on saatu koodattua ja testattua, että se toimii oikein, se voidaan ottaa käyttöön. Käytön yhteydessä sitä ylläpidetään: korjataan virheitä tai uusitaan ja täydennetään sitä.

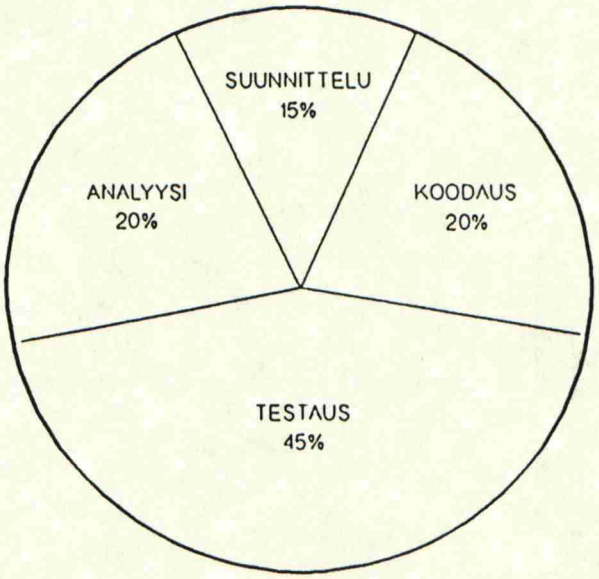
Kuva 12. PERINTEINEN OHJELMISTON ELINKAARI




Lähde: McClure 1989, s. 185

Perinteisesti systeemityö on ollut toteutuspainotteista. Analyysi- ja suunnitteluvaiheisiin on käytetty vain noin 1/3 kehitysprojektien työmäärästä (Kuva 13.). Tämä on johtanut puutteellisesti tai jopa väärin määriteltyihin ohjelmiin.

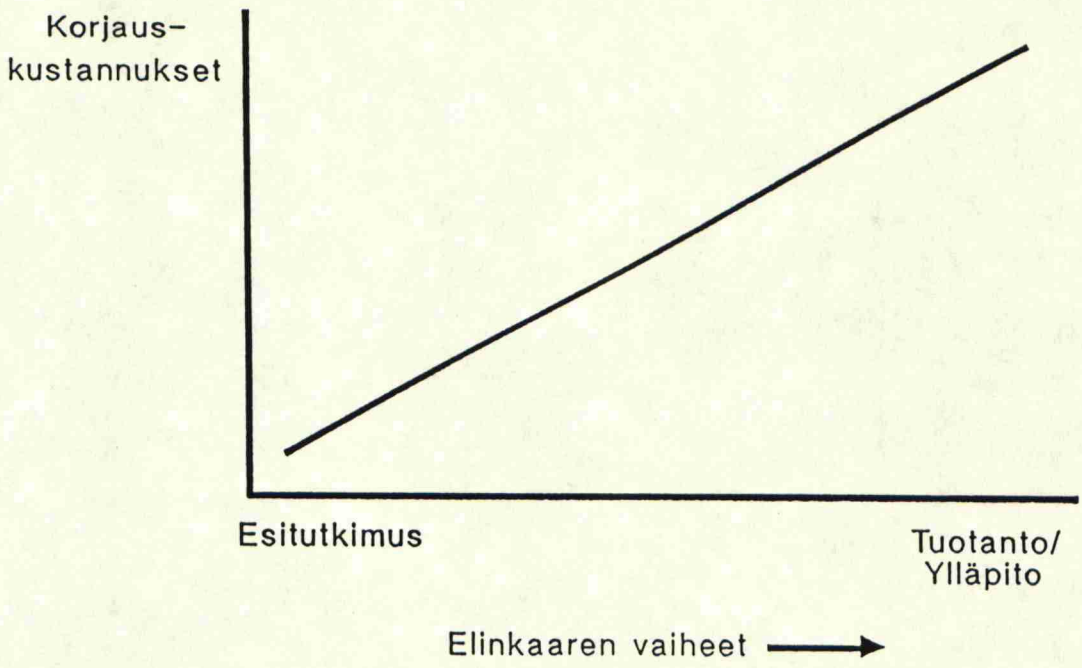
Kuva 13. TYÖMÄÄRIEN JAKAUTUMINEN PERINTEISESSÄ OHJELMISTON ELINKAARESSA




Lähde: McClure 1989, s. 187

Ellei analyysivaiheessa ole tehty protyyppiä järjestelmästä, käyttäjällä on käytännössä ensimmäinen mahdollisuus havaita puutteet ja virheet vasta testausvaiheessa. Tällöin niiden korjaaminen on kuitenkin erittäin työlästä ja kallista - moninkerroin kalliimpaa kuin analyysi- tai suunnitteluvaiheessa (Kuva 14).

Kuva 14. VIRHEIDEN KORJAUSKUSTANNUKSET SYSTEEMITYÖN ERI VAIHEISSA

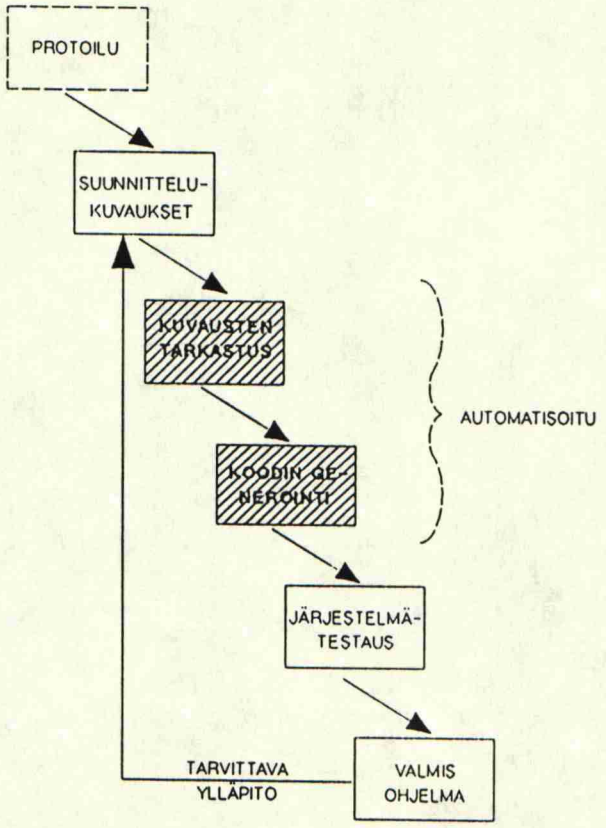



Lähde: Boar 1984, s. 17

TAS-välineiden käyttöönotto muuttaa ohjelmiston elinkaaren vaiheita (Kuva 15.) sekä kehitystyöhön käytettävän työpanoksen jakautumista (Kuva 16.). Manuaalinen koodaus lähes kokonaan poistuu, sillä suunnitteluvaiheen jälkeen saadaan koodigeneraattoreilla generoitua 80-100 prosenttia ohjelmakoodista. Testausvaihe helpottu, koska suurin osa virheistä löydetään automaattisten tarkastusten avulla jo analyysi- ja suunnitteluvaiheissa. Lisäksi koodausvirheet häviävät automaattisen koodigeneroinnin yhteydessä.[1]

[1] McClure 1989 s. 188-190

Kuva 15. OHJELMISTOJEN ELINKAARI TAS:SSÄ

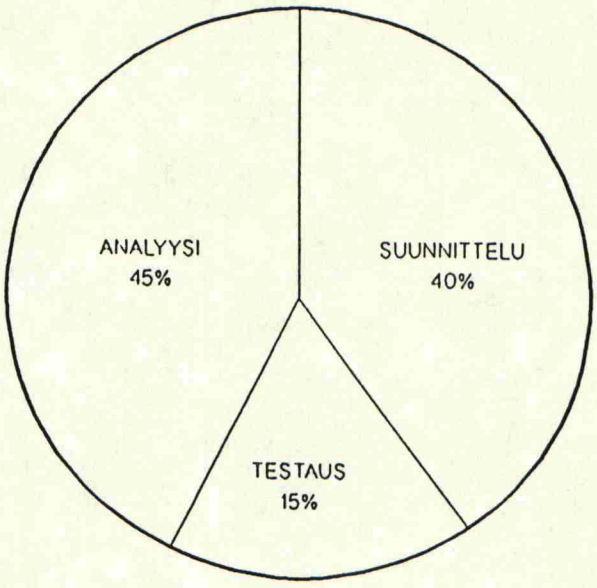



Lähde: McClure 1989, s. 191

Suunnittelijat voivat siis käyttää enemmän aikaa analyysi-ja suunnitteluvaiheissa kuin ennen, koska koodaus- ja testausvaiheisiin ei tarvitse allokoida enää niin paljon aikaa kuin ennen. Analyysivaihessa voidaan tehokkaasti käyttää TAS-välineiden protoiluominaisuuksia, kuten näyttöjen ja raporttien maalausta.[1]

[1] McClure 1989, s. 188

Kuva 16. TYÖMÄÄRIEN JAKAUTUMINEN TAS-VÄLINEITÄ KÄYTETTÄESSÄ




Lähde: McClure 1989, s. 189

## 3.5 Yhteenveto tietokoneavusteisesta systeemityöstä

Tietokoneavusteisella systeemityöllä tarkoitetaan siis systeeminkehitystyötä tietokoneavusteisesti alusta (esitutkimuksesta) loppuun (toteutus ja ylläpito). Vielä ei kuitenkaan markkinoilla ole TAS-välineitä, jotka täydellisesti, alusta loppuun asti, tukisivat systeemityötä. TAS:llä pyritään McCluren mukaan:[1]

1. Vuorovaikutteiseen ja nopeaan systeemityöympäristöön, joka mahdollistaa suunnitteluresurssien tehokkaan kohdentamisen ja jo aikaisessa systeemityövaiheessa tapahtuvan virhetarkastuksen.

---

[1] McClure, 1989 s.14

2. Mahdollisimman monen, nykyisin manuaalisen, kehitys- ja ylläpitotehtävän automatisoinnin.

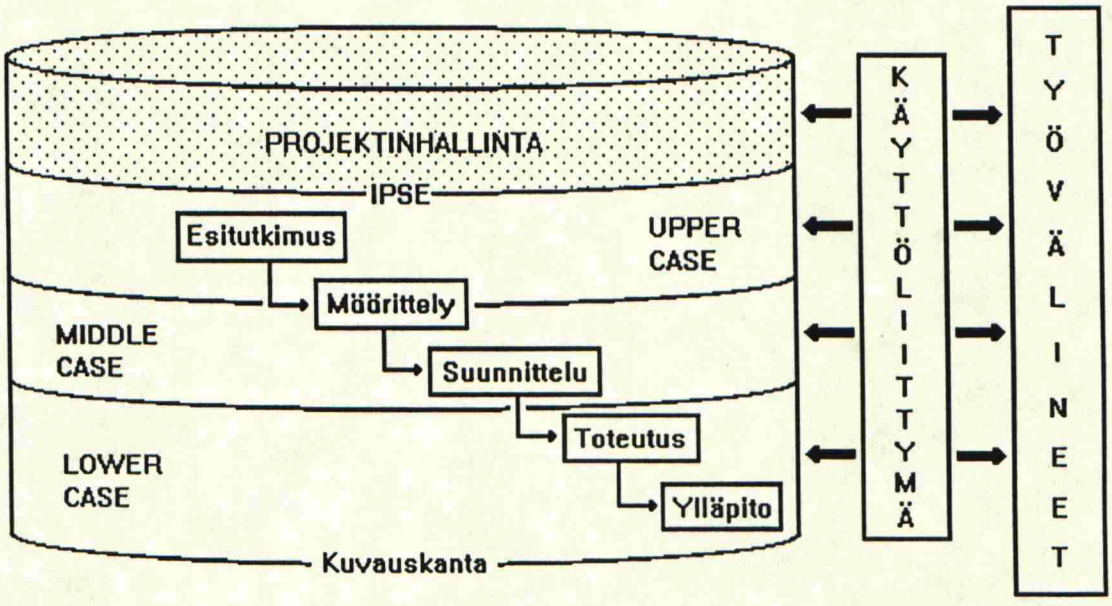
3. Mahdollisuuden visuaaliseen suunnitteluun tehokkaan, graafisen käyttöliittymän avulla.

TAS-ajatteluun liittyy läheisesti myös välineen taustalla oleva menetelmä ja projektienhallinta. Varsinkin systeemityön "kurinalaisuus" ja laatu paranevat, koska TAS-väline pakottaa käyttäjät noudattamaan menetelmäänsä. Projektienhallinta on myös tärkeä osa TAS:tä. Suunnittelutyötä voidaan TAS-välineillä tehdä hajautetusti eri työasemilla, joista kuvaukset ja määritykset talletetaan keskuskuvauskantaan. Projektipäällikön pitäisikin pystyä samaan keskuskuvauskannasta tietoa, kuinka projekti etenee. Tällaisesta TAS-järjestelmästä, johon projektinhallinta on kiinteästi kytketty, käytetään nimitystä IPSE-järjestelmä (IPSE, Integrated Project Support Environment)[1]. Ideaalitapauksessa käytössämme olisi siis TAS-järjestelmä, johon kaikki em. ominaisuudet olisivat integroituina (Kuva 17).

---

[1] McClure, 1989 s. 53

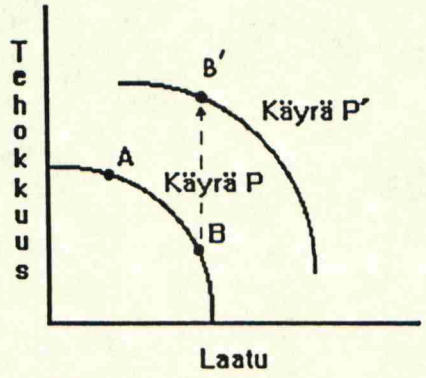
Kuva 17. KOKONAISVALTAINEN TAS-JÄRJESTELMÄ



# 4. TAS-VÄLINEEN VALINTA

Systeemityön tuottavuus muodostuu kahdesta tekijästä: tehokkuudesta ja laadusta. Rakenteisten suunnittelumenetelmien käyttöönotto 70- ja 80-luvuilla paransi tietojärjestelmien laatua, kuitenkin tehokkuuden kustannuksella - suunnittelumenetelmien käyttöönotto hidasti systeemityötä. Yritysten tietojenkäsittelytarpeille syntyi kehitysjättämää (backlog), joka pahimmillaan oli vuosia. 80-luvun puolivälissä markkinoille tulleet TAS-välineet pyrkivät tehostamaan näiden rakenteisten suunnittelumenetelmien käyttöä systeemityössä. Näin pyritään systeemityön tuottavuutta parantamaan (kuva 18).[1]

Kuva 18. SYSTEEMITYÖN TUOTTAVUUSKÄYRÄ JA SEN MUUTOS TAS-VÄLINETTÄ KÄYTETTÄESSÄ




Lähde: Case 1985, s. 36

Rakenteisten suunnittelumenetelmien käyttöönotto ei parantanut systeemityön tuottavuutta. Järjestelmien laatu tosin parani, mutta suunnitteluun kului enemmän aikaa. Kuvassa 19. siirryttiin samalla tuottavuuskäyrällä T pisteestä A pisteeseen B. TAS-välineillä pyritään parantamaan rakenteisten suunnittelumenetelmien käyttöä systeemityön tehokkuuden tai järjestelmien laadun siitä kärsimättä, jolloin tuottavuuskäyrä T siirtyy asemaan T'. Samanlaatuisia järjestelmiä saadaan siis nopeammin kuin ennen (B -> B').

---

[1] Case 1985, s. 36

TAS-välineen käyttöönotto antaa yritykselle mahdollisuuden parantaa systeemityönsä tuottavuutta. TAS-välineen valintaan on syytä kuitenkin paneutua huolellisesti. TAS-välineen hankinnassa tärkeitä huomioitavia asioita ovat:

1. Mikä on yrityksen tietojenkäsittelyn kehittyneisyysaste?
2. Minkälaisia sovelluksia TAS-välineellä on tarkoitus kehittää?
3. Mikä ja minkälainen on käytettävä systeemityömenetelmä?

Yrityksen tietojenkäsittelyn kehitysastetta voidaan arvioida esim. Nolanin mallilla (Nolan 1979), kuten kappaleessa 2.1.1 kuvattiin. Finkelstein esitti mallin pohjalta, minkälaiset TAS-välineet sopivat mihinkin kehitysvaiheeseen. Sen mukaan ei ole esimerkiksi järkevää hankkia yritykseen, jonka tietojenkäsittely on tietohallintovaiheessa, TAS-välinettä, joka on tarkoitettu yksittäisten tietojärjestelmien suunnitteluun ja toteuttamiseen.

TAS-välineellä kehitettävien sovellusten tyyppi ratkaisee myös osaltaan, minkälainen TAS-väline olisi järkevää hankkia. Eräs tapa arvioida sovellustyyppi on käyttää Burns & Dennisin kontingenssimallia (Burns & Dennis, 1985). Kontingenssimalli esiteltiin kappaleessa 2.1.2. TAS-välineet tukevat vielä protoilua heikosti. Tämä seikka vaikuttaa välineen valintaan huomattavasti, varsinkin jos kontingenssimallin mukaan olisi systeemityössä järkevää käyttää prototyyppityöskentelyä. Tällä hetkellä markkinoilla olevat TAS-välineet soveltuvat parhaiten suurten, mutta käsittelylogiikaltaan verrattain yksinkertaisten sovellusten suunnitteluun ja toteuttamiseen.

TAS-välineen valintaan vaikuttaa myös yrityksessä käytössä oleva lähestymistapa systeemityöhön. Lähestymitapa voi olla McCluren mukaan tieto- tai toimintokes-

keinen, kuten kappaleessa 2.2.1 esitettiin. Tietokeskeinen lähestymistapa voi lisäksi keskittyä joko koko yrityksen tietojenkäsittelyn arkkitehtuuriin tai pelkästään yhden sovelluksen tietoihin.

TAS-välineiden tekniset ominaisuudet ovat myös tärkeitä huomionotettavia tekijöitä. TAS-välineen käytön kannalta ovat tärkeimpiä teknisiä ominaisuuksia esiteltiin kappaleessa 3.2 (Finkelstein ja Baram & Steinberg). Teknisten hienouksien ja erikoisuuksien ei saa kuitenkaan antaa vaikuttaa liiaksi TAS-välineen valinnassa. Paljon tärkeämpää on, että väline soveltuu tukemallaan menetelmällään tietyn yrityksen tietynlaisten sovellusten suunnitteluun ja toteutukseen.

Taulukko 5. ERÄIDEN TAS-VÄLINEIDEN OMINAISUUKSIA

| | VÄLINE | | | | |
|---|---|---|---|---|---|
| | IEW | DEFT | IEF | Excelerator | CASE2000 |
| OMINAISUUS | | | | | |
| Menetelmä | IE | SA/SD | IEM | SA/SD | SA/SD |
| Menetelmätuki | X | - | X | - | X |
| Menetelmän sovel-tuvuus suunnit-telussa | koko arkki-tehtuuri | yksitt. sovel-lukset | koko arkki-tehtuuri | yksitt. sovel-lukset | yksitt. sovel-lukset |
| Vaiheet | | | | | |
| -Esitutkimus | X | - | X | - | - |
| -Määrittely | X | X | X | X | X |
| -Suunnittelu | X | X | X | X | X |
| -Tekn. suunn. | X | - | X | - | - |
| -Toteutus | X | - | X | - | - |
| Mallintaminen | | | | | |
| -Tiedot | X | X | X | X | X |
| -Toiminnot | X | X | X | X | X |
| -Näytöt ja raportit | X | X | X | X | X |
| -Keskustelut | - | - | X | - | - |
| Projektinhallinta | - | - | - | - | X |
| Protoilu | - | - | X | X | - |

Lähde: SYTYKE TAS-raportti sekä tekijän omat kokeilut

Taulukkoon 5. on koottu tärkeimpiä ominaisuuksia vain muutamista markkinoilla olevista välineistä, tosin IEW, Excelerator ja CASE2000 ovat myydyimpiä TAS-välineitä maailmassa[1]. TAS-välinettä valittaessa tulisi ainakin em. ominaisuuksia arvioida kunkin yrityksen tietojenkäsittelyn tilan kannalta sekä kehitettävien sovellusten tyypin mukaan. Arviontia helpottaa ja käsitteistää lukujen 2 ja 3 viitekehys.

---

[1] Sytyke TAS-raportti, s. 23

# 5. SYSTEEMITYÖN OPETUS HKKK:SSA

Tietojenkäsittely-aine pääaineena pyrkii antamaan opiskelijoille valmiudet toimia joko atk-alalle erikoistuneissa palveluyrityksissä tai tietotekniikkaa hyväksikäyttävissä yrityksissä systeeminkehityksen erikoistehtävissä (atk-ammatit), taloushallinnon kehittämistehtävissä tai muissa ekonomin tehtävissä, joissa tarvitaan taitoja soveltaa tietotekniikkaa.[1] Tämän tavoitteen saavuttamisessa on systeemityön opetuksella ja sen laadulla keskeinen asema. Systeemityön perusteita opetetaan tällä hetkellä systeeminsuunnittelu-, tiedonhallinta- sekä projektinjohtamiskursseilla. TAS:tä on opetettu kahdella eri kurssilla (keväällä 1989 ja 1990). Sovelluskehittimet -kurssilla opetetaan systeeminsuunnittelua prototyyppityöskentelyn näkökulmasta ja sovelletaan sitä 4. sukupolven ohjelmointivälineillä.

## 5.1. Systeemintyön kurssit HKKK:ssa

### 5.1.1 Systeeminsuunnittelukurssi

Systeeminsuunnittelukurssin tavoitteena on antaa opiskelijalle perustiedot tietojärjestelmien arkkitehtuureista sekä suunnittelutyöstä, sen perusmenetelmistä ja apuvälineistä. Kurssi koostuu tällä hetkellä luennoista (28 h) ja harjoituksista (14 h). Luennoilla käsitellään systeeminsuunnittelun teoriaa, tietojärjestelmien arkkitehtuurikysymyksiä ja sosio-teknisiä vaikutuksia sekä erilaisia tietojärjestelmien kuvaustekniikoita. Harjoitustunneista osa käytetään vierailuluentoihin. Varsinaisia systeeminsuunnittelun harjoituksia on vain 8 tuntia, joilla käsiteltiin neljää eri kuvaustekniikkaa: tietovirta-, ohjelmarakenne-, tieto- ja toimintora-

---

[1] HKKK opinto-opas A. 1988-89, s. 33

kenne-(Jackson) sekä toimintokaavioita. Nämä kuvaustekniikat ovat mielestäni keskeisiä tekniikoita, jotka opiskelijan tulee hallita, mutta niiden harjoitteluun käytetään aivan liian vähän aikaa. Lisäksi luentojen ja harjoitusten keskinäinen yhteys on liian vähäistä. Luennot käsittävät pääasiassa hyvin teoreettista aihepiiriä, kun taas harjoitukset keskittyvät kokonaan muutaman kuvaustekniikan harjoitteluun. Varsinkin tietojenkäsittelyn pääaineopiskelijoille kurssin anti on varmasti liian pieni. Heille pitäisikin järjestää jonkinlainen 'systeeminsuunnittelun jatkokurssi', jossa kuvaustekniikoiden käyttöä harjoiteltaisiin lisää. Kurssin teoriaopetusta voisi mielestäni myös hieman supistaa ja painottaa enemmän harjoituksia. Teoriasisällön opiskelu tulisi jättää enemmän opiskelijoiden omalla ajalla tapahtuvaksi. Korkeakouluympäristössä ei liene järkevää käyttää rajallisia opetustunteja siihen, että kurssin teoriamateriaalia käydään lähes sivu sivulta lävitse luennoitsijan johdolla.

### 5.1.2 Tiedonhallintakurssi

Tiedonhallintakurssin tavoitteena on antaa opiskelijalle valmiudet tiedon mallintamiseen ja analysoimiseen. Kurssilla käsitellään kohdeanalyysiä ja relaatiomallia. Kurssi koostuu luennoista (28 h) ja harjoituksista (14 h). Tällä hetkellä kurssi on täysin irrallaan systeeminsuunnittelukurssista. Käytännön systeemityössä kuitenkin tiedonhallinta ja varsinainen systeeminsuunnittelu ovat läheisesti yhteydessä toisiinsa. Systeemisuunnittelukurssin tulisikin mielestäni havainnollistaa opiskelijoille tieto-ja toimintoanalyysin välistä yhteyttä. Tämä vaatisi, että Tiedonhallintakurssi suoritettaisiin ennen Systeeminsuunnittelun kurssia.

### 5.1.3 Projektin johtamiskurssi

Systeemityön opetukseen liittyy myös projektin johtamiskurssi. Kurssin tavoitteena on muodostaa opiskelijalle kuva projektitoiminnasta osana yrityksen liiketoimintaa ja sen strategioiden toteutusta. Tietojenkäsittelyn kurssina projektin johtaminen tulisi myös nähdä systeemikehitystyöhön kiinteästi liittyvänä osana. Tällä hetkellä kurssi keskittyy liiaksi yleiseen projektien johtamiseen ja projektinhallinnan termistön opetteluun, sen sijaan että käsiteltäisiin ATK-projektien johtamista ja sen erikoispiirteiden hallitsemista.

### 5.1.4 Sovelluskehittimet-kurssi

Kurssin tavoitteena on perehdyttää opiskelijat sovelluskehityksen automaattisiin apuvälineisiin, sovelluskehittimiin sekä antaa opiskelijalle valmiudet arvioida niiden soveltuvuutta erilaisiin tehtäviin. Kurssi koostuu luennoista (28 h) ja harjoituksista (28 h) sekä pienryhmätyöskentelystä. Kurssilla tutustutaan prototyyppityöskentelyn teoriaan osana systeeminsuunnittelua, sen taloudellisiin vaikutuksiin, työvälineisiin (sovelluskehittimet) ja työmenetelmiin. Harjoitustyönään opiskelijat toteuttavat jollain sovelluskehittimellä toimivan sovelluksen. Kurssi on hyvin harjoitustyöpainoteinen. Suurin osa kurssiin käytettävästä työmäärästä kuluu harjoitustyön tekemiseen. Vastaava suhde luentojen ja harjoitusten välillä olisi hyvä myös muussa systeemityön opetuksessa.

Sovelluskehittimet-kurssi on mielestäni tällä hetkellä kuitenkin liian irrallinen muusta systeemityön opetuksesta. Kurssin prototyyppityöskentelypainotteinen harjoittelu tulisi läheisemmin kytkeä osaksi systeeminsuunnittelua. Muutoin on vaarana,

että opiskelijat eivät hahmota prototyyppityöskentelyä keskeisenä osatekniikkana tietosysteemien kehittämisessä. Kuten Baram & Steinberg[1] esittävät, käytännössä harvemmin toteutetaan tietosysteemi joko pelkästään ns. perinteisellä tavalla vesiputousmallin mukaisesti tai pelkästään protoilemalla. Yleensä käytetään molempia menetelmiä toisiaan täydentämässä (sekamalli). Tämä asia pitäisi ottaa huomioon myös opetuksessa.

### 5.1.5 Systeeminsuunnittelun CASE-kurssi

Keväällä 1989 aloitettiin HKKK:ssa tietokoneavusteisen systeemityön opetus. Kurssilla pyrittiin integroimaan systeeminsuunnittelu-, tiedonhallinta- ja projektinjohtamiskurssin opit yhtenäiseksi atk-kehitysprojektiksi (Kuva 19.). Tavoitteeeseen ei kuitenkin päästy, koska suurin osa kurssiin varatuista resursseista kului itse tietokoneavusteisten systeemityövälineiden käytön opetteluun. Varsinainen ATK-kehitysprojekti jäi toteuttamatta. Kurssia jouduttiinkin muuttamaan siten, että kurssilaiset jaettiin ryhmiin, jotka tutkivat tietokonenavusteisen systeemityötä ja siihen läheisesti liityviä asioita, kuten liittymiä muihin kehitysvälineisiin, projektityöhön ja kehityskustannusten arviointimenetelmiin.

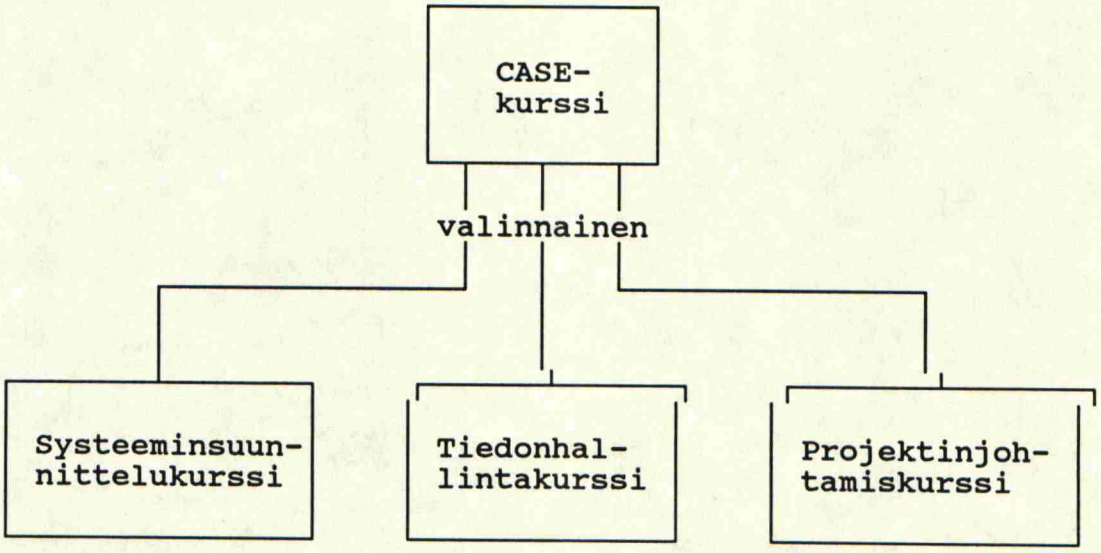
Keväällä 1990 systeeminsuunnittelun CASE-kurssi järjestettiin edellisvuodesta poiketen enemmän systeeminsuunnittelusuuntautuneena, jolla pyrittiin saavuttamaan kurssin oppimistavoitteet aikaisempaa paremmin. Kurssi koostui sekä teoria-, että harjoittelujaksosta. Teoriajakso koostui pääosin tässäkin tutkielmassa aiemmin esitetystä materiaalista. Harjoitusjaksolla tehtiin kuvaukset vähittäiskaupan ostoreskontrasta. Kuvakset tehtiin sekä tieto- että

---

[1] Baram & Steinberg, 1989

toimintorakenteista. Lisäksi tehtiin projektisuunnitelmat siitä kuinka suunnitteluprojekti etenee ja koska se valmistuu kunkin ryhmän käytössään olevilla resursseilla. Suunnitteludokumentit tehtiin joko DEFT:llä tai IEW:llä työryhmän sisäisen valinnan perusteella. Tälläkin kurssilla suurimpana ongelmana oli itse TAS-välineiden käytön opettelu. Varsinkin IEW:n perusteiden opetteluun kului liikaa aikaa. DEFT:n oppimiskynnys oli selvästi IEW:tä alempi.

Kuva 19. SYSTEEMINSUUNNITTELUN CASE-KURSSI OPETUSTA INTEGROIVANA KURSSINA




## 5.2 Systeemityön opetuksen ongelmat ja parannusehdotukset

Tällä hetkellä ongelmana opetuksessa on se, että jollei opiskelija suorita valinnaista systeeminsuunnittelun CASE-kurssia tai jollei CASE-kurssi saavuta sille asetettuja tavoitteita, niin systeemityön kokonaisuus saattaa jäädä hänelle hahmottumatta. Zachmanin[1] tietojärjestelmän arkkitehtuurinäkemystä systeemityöstä erilaisten ja eritasoisten suunnittelukuvausten integroimisena tulisi korostaa opetuksessa. Mielestäni järkevämpää olisikin tehdä systeeminsuunnittelun CASE-kurssista tiedon- ja projektinhallinnan sekä systeeminsuunnittelumenetelmien integrointikurssi (Kuva 20.). Se olisi samalla tietojenkäsittelyn pääaineopiskelijoille pakollinen 'systeeminsuunnittelun jatkokurssi'.

Tämä vaatisi kaikkien kolmen em. kurssin sisällön ja harjoitusten yhtenäistämistä. Kurssien harjoitustöissä paneuduttaisiin samaan systeeminkehitysprojektiin kunkin kurssin näkökulmasta. Systeeminsuunnittelun

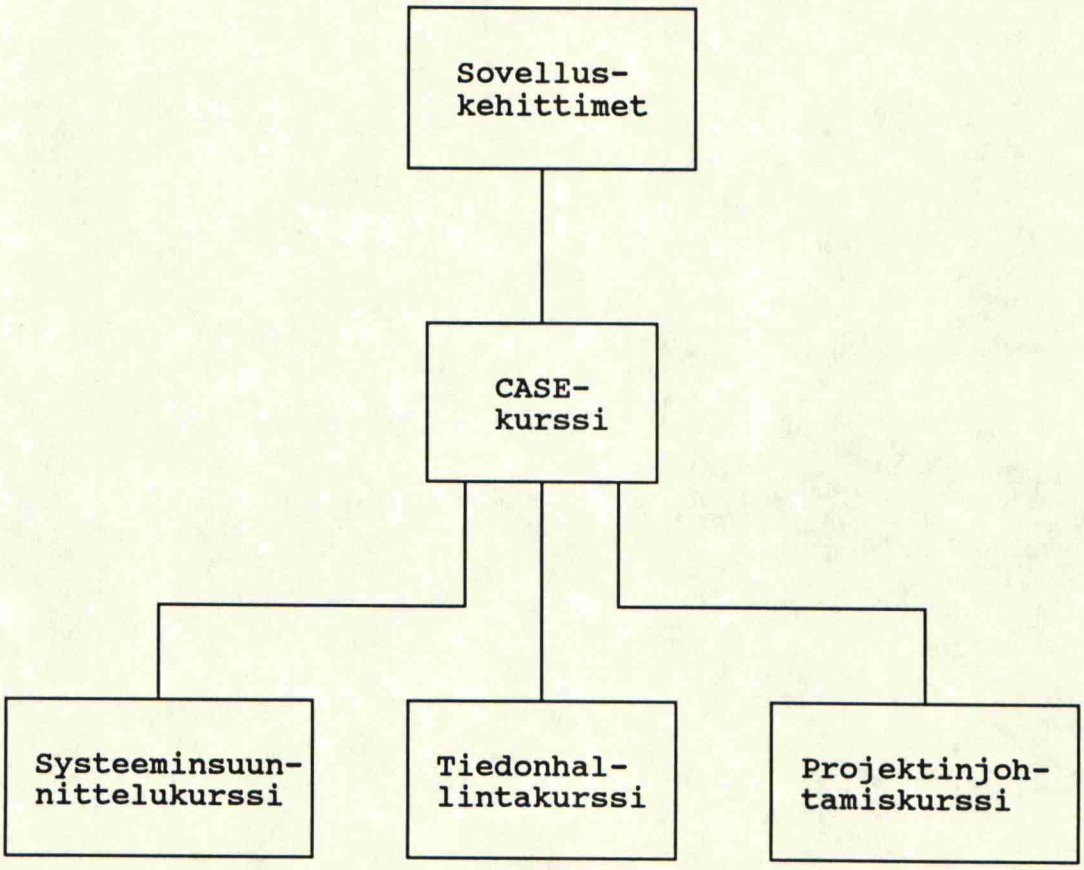
---

[1] Zachman, 1987, s 276-292

CASE-kurssilla käytettäisiin edelleen samaa harjoitustyötä, kuin em. kursseillakin. Nyt vain tehtäisiin tietokoneavusteisilla työvälineillä täydellisemmät kuvaukset opituilla tekniikoilla myöhempää Sovelluskehittimet -kurssilla tapahtuvaa toteutusta varten. CASE-kurssi, kuten myöhemmin Sovelluskehittimet-kurssikin, toteutettaisiin ennalta suunniteltuna projektina Projektin johtamiskurssin oppien mukaisesti. Näin varsinkin pääaineopiskelijat saisivat kokonaisnäkemyksen systeeminsuunnittelun tekniikoista ja työtavoista. TAS-välineeksi kurssille sopisi DEFT IEW:tä paremmin, koska sen käytön opetteluun kuluu huomattavasti vähemmän aikaa, jolloin varsinaisille harjoitustöille jää enemmän aikaa.

Sovelluskehittimet-kurssilla voitaisiin koko systeeminsuunnittelun opetuksen päätökseksi totetuttaa tietojärjestelmä, joka on suunniteltu aikaisemmilla kursseilla. Ehdotettu kurssirakenne antaisi pääaineopiskelijoille nykyistä paremman kuvan systeemityöstä kokonaisuudessaan. Tällaista kokonaisuutta ei pystytä toteuttamaan yhdellä kurssilla, joten kaikkien systeeminsuunnittelun kurssien integroiminen yhtenäiseksi suunnittelu- ja toteutusprojektiksi on ainoa mahdollisuus.

Kuva 20. EHDOTUS KURSSIRAKENTEEKSI HKKK:N SYSTEEMIN-SUUNNITTELUN OPETUKSEEN




Kuvan 20. kurssit tulisivat olla tietojenkäsittelyaineen pääaineopiskelijoille pakollisia. Opinnot voisi ajoittaa esimerkiksi seuraavasti:

Pääaineopintojen 1. vuoden syksy: Tiedonhallinta
1. vuoden kevät: Systeeminsuunnittelu
Projektinjohtaminen
2. vuoden syksy: CASE-kurssi
2. vuoden kevät: Sovelluskehittimet

Projektin johtamiskurssi olisi mielestäni järkevintä suorittaa Tiedonhallinta- ja Systeeminsuunnittelukurssin jälkeen, jolloin opiskelijoilla on jo tietty näkemys systeemityöstä ja opetuksessa voidaan näin ollen keskittyä systeemityöprojektien suunnitteluun ja valvontaan. Esimerkiksi voitaisiin suunnitella, kuinka CASE- ja Sovelluskehittimet kursseilla myöhemmin suunnitellaan ja toteutetaan tietty tietojärjestelmä. Näillä kursseilla pienryhmät

toteuttaisivat kukin osan järjestelmästä opettajan toimiessa projektipäällikkönä.

Kuvatunlainen kurssirakenne ja -integraatio vaatii aluksi luennoitsijoilta entistä läheisempää yhteistyötä ja keskinäistä koordinointia. Myöhemmin, kun selkeä yhteinen luento- ja harjoituskokonaisuus on saatu suunniteltua ja toteutettua, tämä ratkaisu ei tuottane ylimääräistä työtä opettajille.

# 6. PÄÄTELMÄT JA JATKOTUTKIMUSAIHEET

TAS-välineellä on mahdollisuus parantaa yrityksen systeemityön tuottavuutta merkittävästi. Tuottavuusedut eivät välttämättä paljastu ensimmäisten projektien myötä, mutta myöhemmin etenkin kuvausten muokattavuus ja uudelleenkäyttö nopeuttavat systeemityötä ja parantavat myös sen laatua.

TAS-järjestelmän tulisi kattaa kaikki systeemityön vaiheet esitutkimuksesta käyttöönottoon ja ylläpitoon. Tällä hetkellä ei markkinoilla ole tuotetta, joka täysin kattaisi kaikki systeemityön osa-alueet. Suurimmat puutteet näissä parhaissakin välineissä ovat projektihallinnan puolella-se joko puuttuu tai on puutteellinen.

Tällä hetkellä markkinoilla olevat TAS-välineet ovat täysin ATK-ammattilaisten työvalineitä. TAS-välineisiin tullaan kuitenkin sisällyttämään yhä enemmän tekoälyä, joka ohjaa niiden käyttöä teknisesti ja menetelmällisesti. Tällöin myös itsenäiskäyttäjillä on mahdollisuus suunnitella ja toteuttaa tietojärjestelmiä. Myös takaisinsuunnitteluominaisuudet (reverse engineering) tulevat parantumaan. Tämä nopeuttaa huomattavasti vanhojen ohjelmien ylläpitotyötä.

TAS-välineiden käyttöönotto muuttaa systeemien kehitykseen kuluvan työajan ja -määrän jakautumista. TAS-välineitä käytettäessä suurin osa systeemien kehityksen työmäärästä käytetään analyysi- ja suunnitteluvaiheissa, kun taas perinteisessä systeemityössä suurin osa kokonaistyömäärästä kuluu koodaus- ja testausvaiheissa.

Systeeminsuunnittelun opetus HKKK:ssa on tällä hetkellä liian paloittaista. Nykyisin systeemityössä korostetaan kokonaisvaltaista systeeminsuunnittelua, jossa suunnittelutyö alkaa yrityksen strategioista ja päämääristä. Tietojärjestelmäarkkitehtuuri johdetaan näistä lähtökohdista yrityksen tarpeiden mukaan. Tällaista suunnittelua tulisi korostaa myös opetuksessa. Tällä hetkellä systeeminsuunnittelun kurssit eivät riittävän hyvin tue tietojärjestelmäark-

kitehtuurien suunnittelua ja toteutusta. Niiden sisältöä pitäisikin yhtenäistää suuremmaksi systeeminsuunnitteluprojekteiksi, jolloin pääaineopiskelijat saavat riitävän kattavan kuvan systeemityöstä kokonaisuudessaan.

**Jatkotutkimusaiheita:**

- TAS-välineen käyttöönoton vaikutukset yrityksen systeemityökulttuuriin: muuttuvatko systeemityöntekijöiden työtehtävät.

- TAS-välineen käyttöönoton vaikutukset systeemityön tuottavuuteen. Saavutetaanko niitä ja kuinka paljon? Kuinka kaun kestää ennen kuin mahdolliset tuottavuushyödyt alkavat näkymään?

- Kuinka on käynyt sellaisissa yrityksissä, jotka ovat valinneet tämän tutkielman viitekehyksen mukaan vääränlaisen välineen. Onko systeemityön tuottavuus parantunut vai huonontunut tällaisessa yrityksessä vai onko mahdollisesti TAS-väline jäänyt kokonaan käyttämättä.

- Paraneeko vai huononeeko systeemityöprojektien hallittavuus käytettäessä TAS-välineitä?

LÄHTEET

Acly, Ed

Computer-Aided Software Engineering (CASE). The Meaningless Term to Describe a Very Meaningfull Technology. International Data Corporation Bulletin (#3244 October 27, 1987)

ATK-Sanakirja

Suomen ATK-kustannus Oy, Espoo, 1987.

Baram, G., Steinberg G.

"Selection Criteria for Analysis and Design CASE Tools". ACM SIGSOFT, Software Engineering Notes, vol 14 no 6, oct. 1989.

Boar, Bernard

Application Prototyping. A Requirements Definition Strategy for the 80's. John Wiley / Sons inc., New York 1984.

Boehm B.

"Software Engineering Economics". Prentice-Hall, New Jersey 1981.

Burkhard, D. L, Jenster, P.V.

Applications of Computer-Aided Software Engineering Tools: Survey of Current and Prospective Users. DATA BASE, Fall 1989, pp. 28-37.

Burns R.N., Dennis A.R.

Selecting the Appropriate Application Development Methodology. DATA BASE, Volume 17, No. 1, Fall 1985, pp. 19-23.

Case Albert F. Jr.

Computer-Aided Software Engineering (CASE): Technology for Improving Software Engineering Development Productivity. DATA BASE, Volume 17, No. 1, Fall 1985. pp. 35-43.

CASE Outlook

Lehden numerot: October 1987 - March 1988.

Charette, Robert
Software Engineering Environments - Consepts and Technology. McGraw-Hill, New York 1986.

DeMarco, T.
Structured Analysis and Systems Specifications. Prentice-Hall, Englewood Cliffs, N.J., 1979.

Finkelstein, Clive
CASE Methodologies, Tools and Techniques. CASExpo, Washington 19.-20.10.1987.

Gane, C., Sarson, T.
Structured Systems Analysis. Prentice-Hall, Englewood Cliffs, N.J., 1979.

Gibson, Michael
The CASE Philosophy. BYTE, April 1989.

Margolis, Nell
CASE Systems Handles Reverse Engineering. Computerworld, August 29. 1988.

Martin, Charles
Second-generation CASE Tools: A Challenge to Vendors. IEEE Software, March 1988.

Martin, James
Application Development without Programmers. Prentice-Hall, New Jersey 1982.

Martin, James & McClure, Carma
Structured Techniques: The Basis of CASE. Prentice-Hall, New Jersey 1988.

McClure, Carma
CASE is Software Automation. Prentice-Hall, New Jersey 1989.

The CASE for Structured Techniques. PC Tech Journal, vol.6 no.8, August 1988.

The CASE experience. BYTE, April 1989.

Nolan, R. L.
Managing the Crises in Data Processing. Harvard Business Review. March-April, 1979. pp. 115-126.

Peters, Lawrence J.
Software Design: Methods and Techniques. Yourdon Press, New York, 1981.

Systeemityöyhdistys, SYTYKE ry.
TAS-projektin raportti: Systeeminsuunnittelun Työasema. Suomen ATK-kustannus Oy, Espoo 1988.

Yourdon, E.
Modern Structured Analysis. Prentice-Hall, Englewood Cliffs, N.J., 1979.

Zachman, J.A.
A Framework for Information Systems Architechture. IBM Systems Journal, vol.26 no.3 1987.